Aim : Identify Functional and Non-Functional Requirements for:

(i) Online Ticket Reservation for Railways

(ii) Online Auction Sales

Procedure :

Online Ticket Reservation for Railways

Functional Requirements

- Every online booking needs to be associated with an account
- One account cannot be associated with multiple users
- Search results should enable users to find the most recent and relevant booking options
- System should enable users to book / pay for their tickets only in a time boxed manner after tickets being added to the cart
- System should only allow users to move to payment only when mandatory fields such as date, time, location has been mentioned
- System should consider time zone synchronization when accepting bookings from different time zones
- Booking confirmation should be sent to user to the specified contact details

Non Functional Requirements

- Use of captcha and encryption to avoid bots from booking tickets
- Search results should populate within acceptable time limits
- User should be helped appropriately to fill in the mandatory fields, incase of invalid input
- System should accept payments via different payment methods, like PayPal, wallets, cards, vouchers, etc
- System should visually confirm as well as send booking confirmation to the user's contact

Online Auction Sales

Functional Requirements

- Only users with administrative login can access the control admin page and have full access right to the system
- Basic account users can only view sales and bid
- Basic account users are recognized by the strength of their bought items and sellers feedbacks
- Seller account users can only sell items and have to pay a subscription fee
- Seller account users are recognized by how many positive feedbacks and stars a seller has
- Advanced account users can only sell, buy or trade items, advanced account users are recognized by how well is their feedback from both sellers and buyers on the auction system, i.e. advanced account member that has 120 sales to his/her account with all positive feedbacks and stars then this seller/trader is very good
- Allow visitors to sign up for account online, Login and Logout to the system options
- New members have the option to upgrade to seller or advanced account
- Admin user can create accounts and delete accounts
- Basic account users can contact a seller privately
- Sellers can create their own discussion forum for the item on sale
- Enabling a reminder on an item for when its soon to finish
- Users can pay online using a secure payment system
- Sellers can have flash images or videos to add to their advertisement
- Guests can view and search for items but not be able to bid unless they sign up for an account
- Admin members can change sale status, Basic, Seller and Advanced members can delete or edit their account/profile

Non Functional Requirements

- Ease of navigation around the site and use of simple colors and fonts on the system
- All web pages that are generated throughout the website should be downloadable in no more than 10 seconds over 40KBps modem connection.
- Interface consistency throughout the system.
- The response to a query shouldn't take a very long time and not more than 10 seconds to load on screen.
- When a user sells or bids on the system they should be getting a confirmation of what the member did, i.e. adding a new item on sale at the end of the adding to the system they should get a conformation message to say its been added

Result : **Thus we have identified functional and non functional requirements of online auction sales and online ticket reservation system.**

Aim : Construct a flow graph for Insertion sort algorithm.

Procedure :

Cyclomatic Complexity

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the

graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

For example, if source code contains no control flow statement then its cyclomatic complexity will be 1 and source code contains a single path in it. Similarly, if the source code contains one if condition then cyclomatic complexity will be 2 because there will be two paths one for true and the other for false.

Mathematically, for a structured program, the directed graph inside control flow is the edge joining two basic blocks of the program as control may pass from first to second.

So, cyclomatic complexity M would be defined as

M = E – N + 2

where,

E = the number of edges in the control flow graph

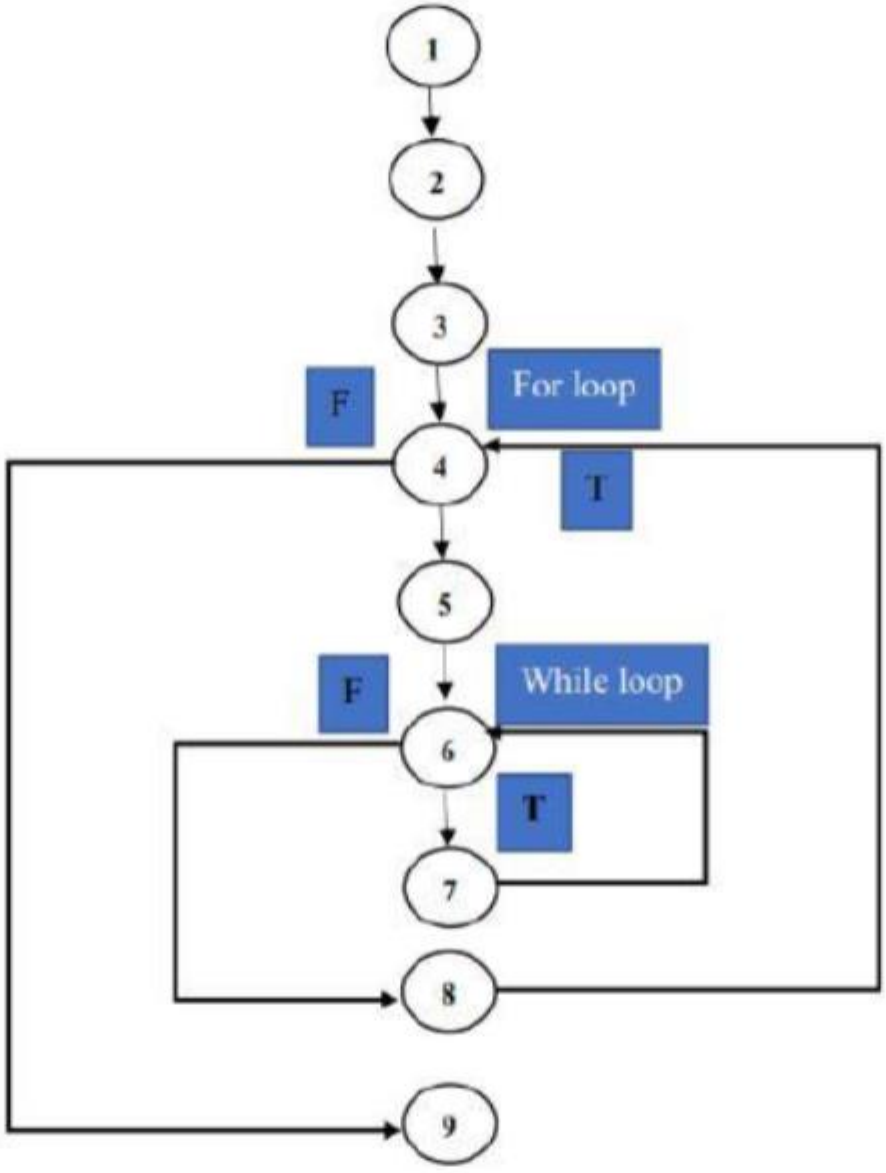
N = the number of nodes in the control flow graph

P = the number of connected components

The following steps should be followed for computing Cyclomatic complexity and test cases design.

Step 1 - Construction of graph with nodes and edges from the code

Flow Graph for Insertion Sort Algorithm



Step 2 – Identification of independent paths

Step 3 – Cyclomatic Complexity Calculation

Step 4 – Design of Test Cases

Once the basic set is formed, TEST CASES should be written to execute all the paths.

Insertion Sort Algorithm

```
def InsertionSort(A, n) {
    for i = 0 to n {
        k = A[i];
        j = i – 1;
        while (j >= 0) and (A[j] > k) {
            A[j + 1] = A[j];
            j = j – 1;
        }
        A[j + 1] = k;
    }
    print(A);
    A = [11, 12, 13, 14];
    InsertionSort(A, len(A));
}
```

Result : Thus the flow graph of Insertion sort has been drawn.

Aim : Write a program to find Cyclomatic complexity for the above flow graph.

Procedure :

Number of edges : 10

Number of nodes : 9

Cyclomatic complexity : 3

Manual Calculation of Cyclomatic Complexity for Insertion Sort

Mathematically, it is set of Independent paths through the graph diagram. The Code complexity of the program can be defined using the formula –

V(G) = E – N + 2

Where,

E – Number of Edges

N – Number of Nodes

V(G) = P + 1

Where P – Number of predicate nodes (node that contains condition)

- V(G) = E – N + 2 = 10 – 9 + 2 = 3
- V(G) = P + 1 = 2 + 1 = 3 (Condition nodes are 4 and 6 nodes)

Use of Cyclomatic Complexity :

- Determining the independent path executions thus proven to be very helpful for Developers and Testers.
- It can make sure that every path have been tested at least once.
- Thus help to focus more on uncovered paths.
- Code coverage can be improved.
- Risk associated with program can be evaluated.
- These metric being used earlier in the program helps in reducing the risks.

Advantages of Cyclomatic Complexity :

- It can be used as a quality metric, gives relative complexity of various designs.
- It is able to compute faster than the Halstead's metrics.
- It is used to measure the minimum effort and best areas of concentration for testing.
- It is able to guide the testing process.
- It is easy to apply.

Disadvantages of Cyclomatic Complexity :

- It is the measure of the programs' control complexity and not the data the data complexity.
- In this, nested conditional structures are harder to understand than non-nested structures.
- In case of simple comparisons and decision strucures, it may give a misleading figure.

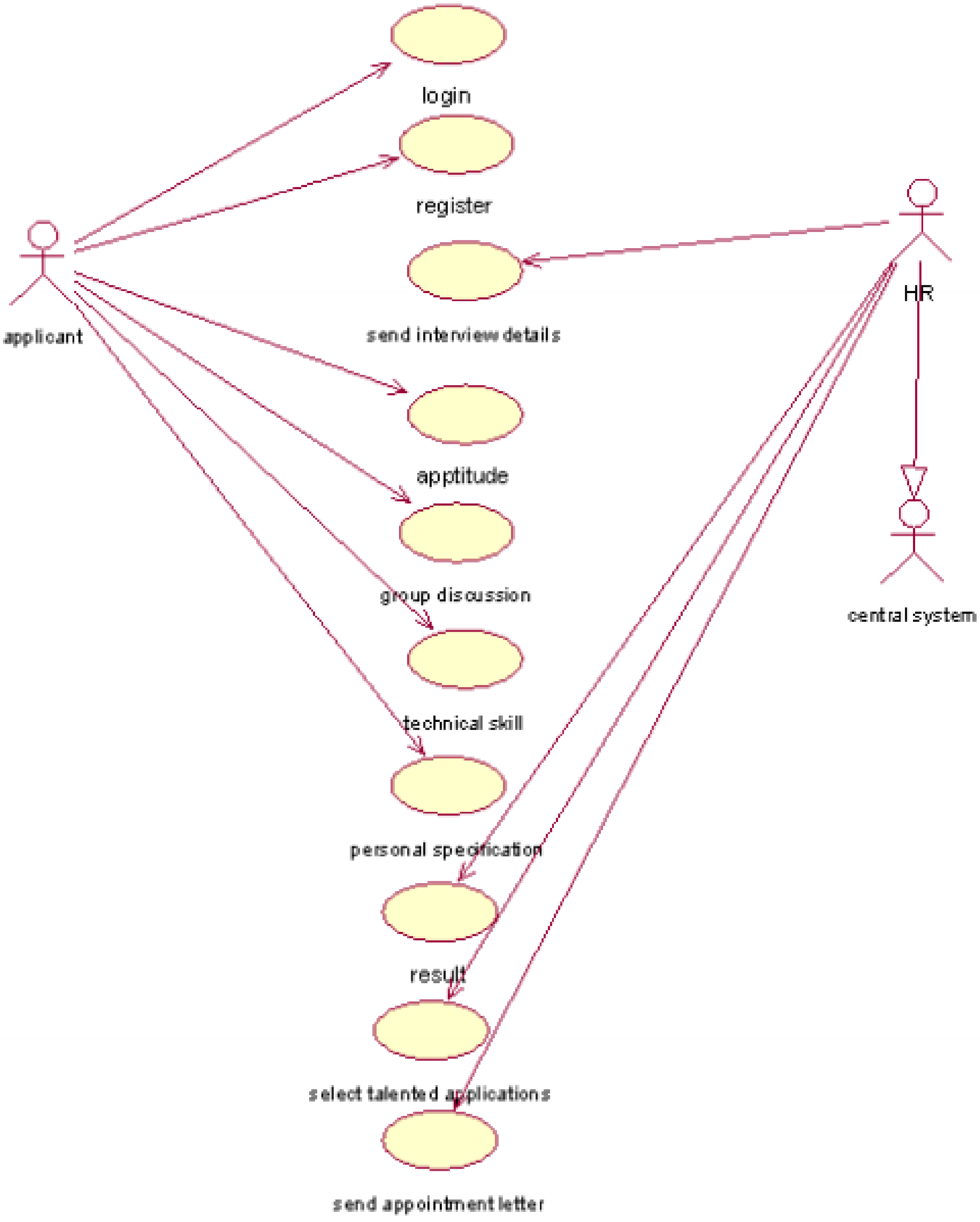
Result : Thus the cyclomatic complexity of the above flow graph has been computed.

Aim : To model all the views of Recruitment Procedure for Software Industry

Problem Statement :

In the software industry the recruitment procedure is the basic thing that goes in the hand with the requirement as specified by the technical management team. HR first gives an advertisement in leading Newspapers, Journals, Weekies and Websites. The job seekers can apply for it through by Post or by e-mail to the company. The technical skill and the experience of the candidates are reviewed and the short listed candidates are called for the interview. There may be different rounds for interview like the written test, technical interview, and HR interview .Afte the successful completion of all rounds of interview, the selected candidates' names are displayed. Meanwhile HR gives all the details about the salary, working hours, terms and conditions and the retirement benefit to the candidate.

Usecase Diagram



USECASE VIEW

- Uscase diagrams model the functionality of a system using actors and usecases which are a set of sequence of actions.

Usecases are

- login
- register
- send interview details
- aptitude
- group discussion
- technical skill
- personal specification
- result
- select talented applications
- send appointment letter

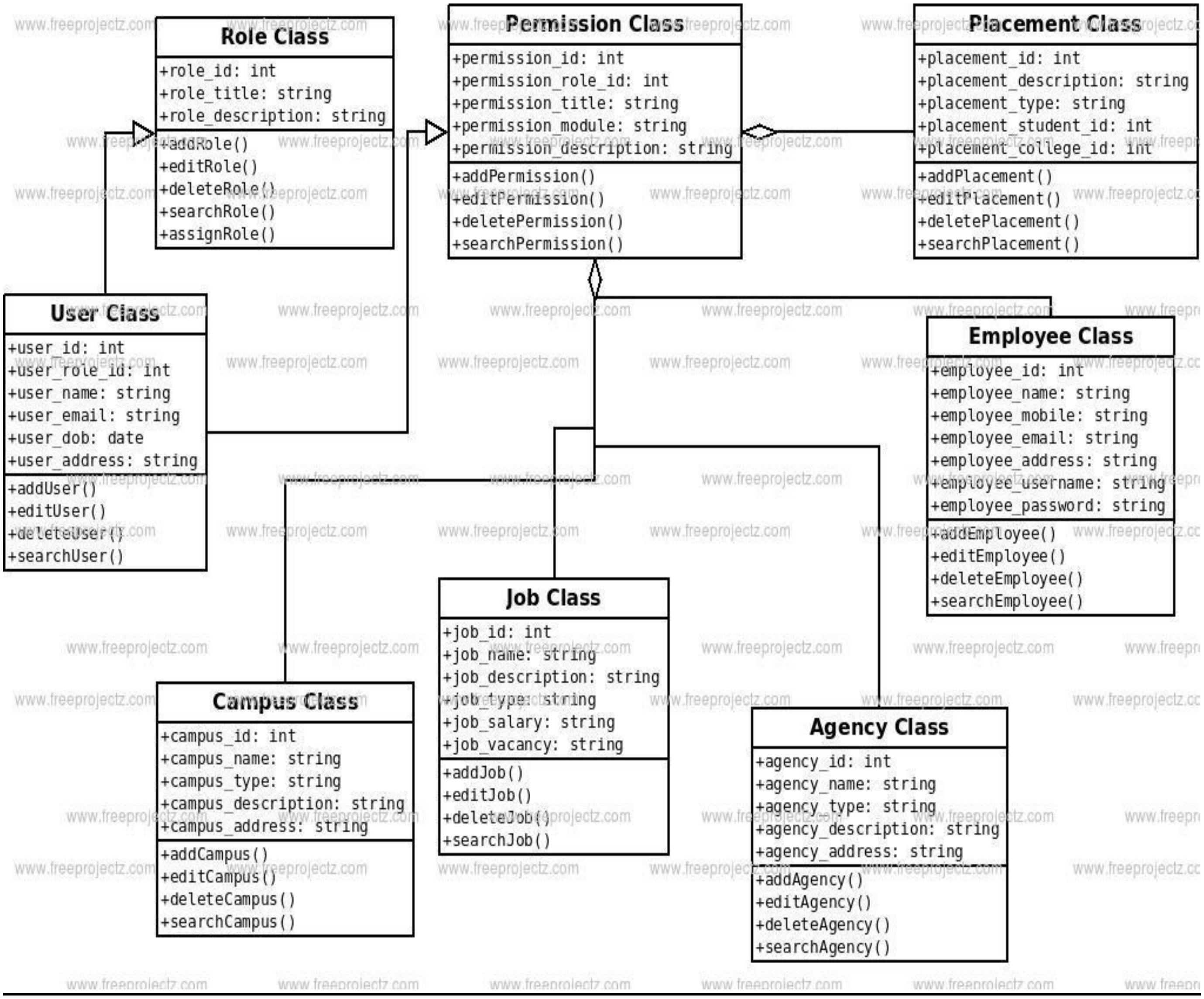
Actors are

- Applicant
- HR
- Central system

Relationships are

- Association
- Generalization

Class Diagram



CLASS DIAGRAM

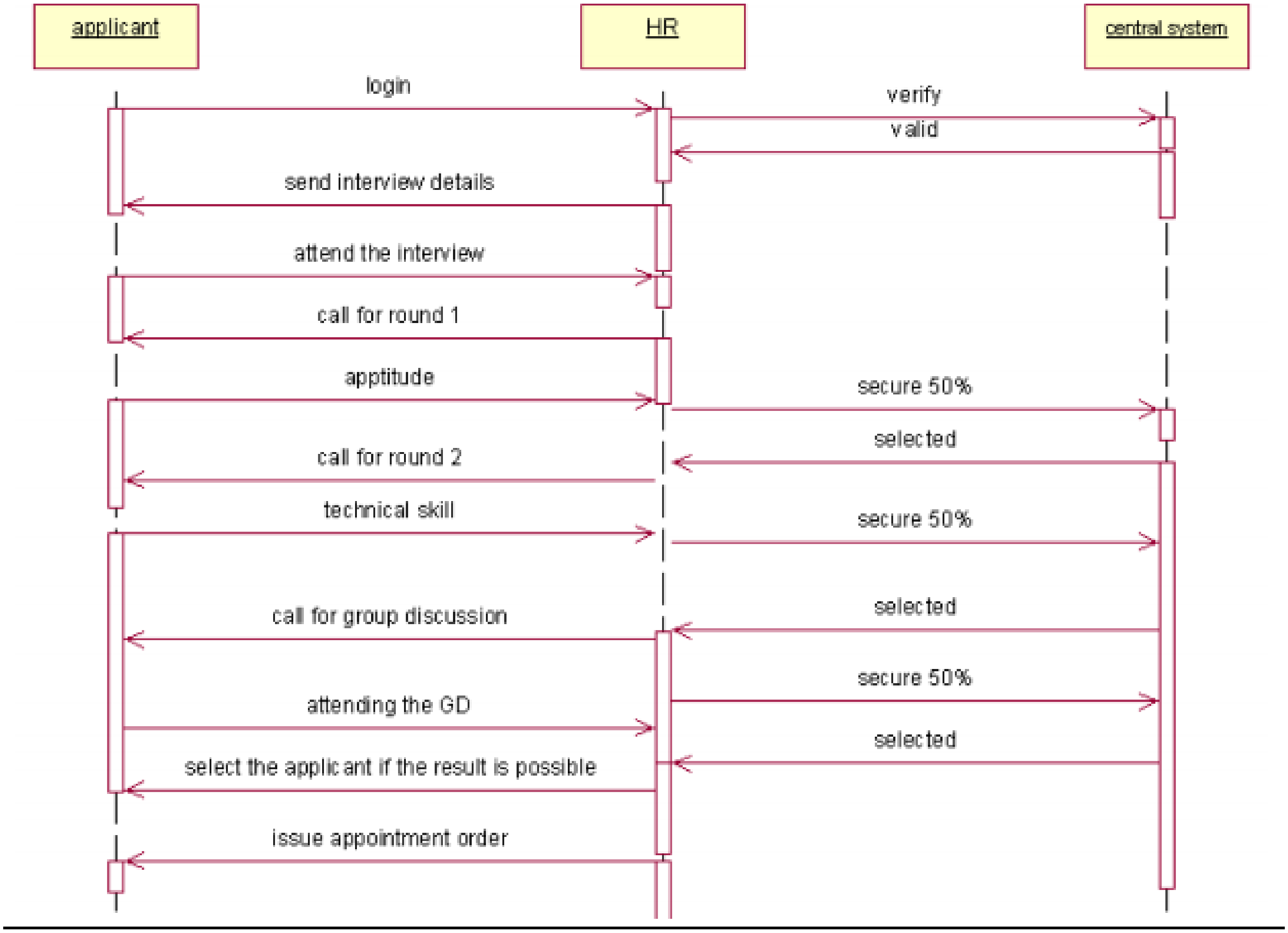
- Class diagram describes the structure of a system.

S.No	Class Name	Attributes	Operations
1.	Role Class	role_id, role_title, role_description	addRole(), deleteRole()
2.	Permission Class	permission_id, permission_description	addPermission(), deletePermission()
3.	Placement Class	placement_id, placement_description	addPlacement(), deletePlacement()
4.	Employee Class	employee_id, employee_details	addEmployee(), deleteEmployee()
5.	User class	user_id, user_details	addUser(), deleteUser()
6.	Campus class	campus_id, campus_details	addCampus(), deleteCampus()
7.	Job class	job_id, job_description	addJob(), deleteJob()
8.	Agency class	agency_id, agency_description	addAgency(), deleteAgency()

The Relationships are

- Association
- Dependency
- Generalization

Sequence Diagram



SEQUENCE DIAGRAM

- It gives time ordering of messages.

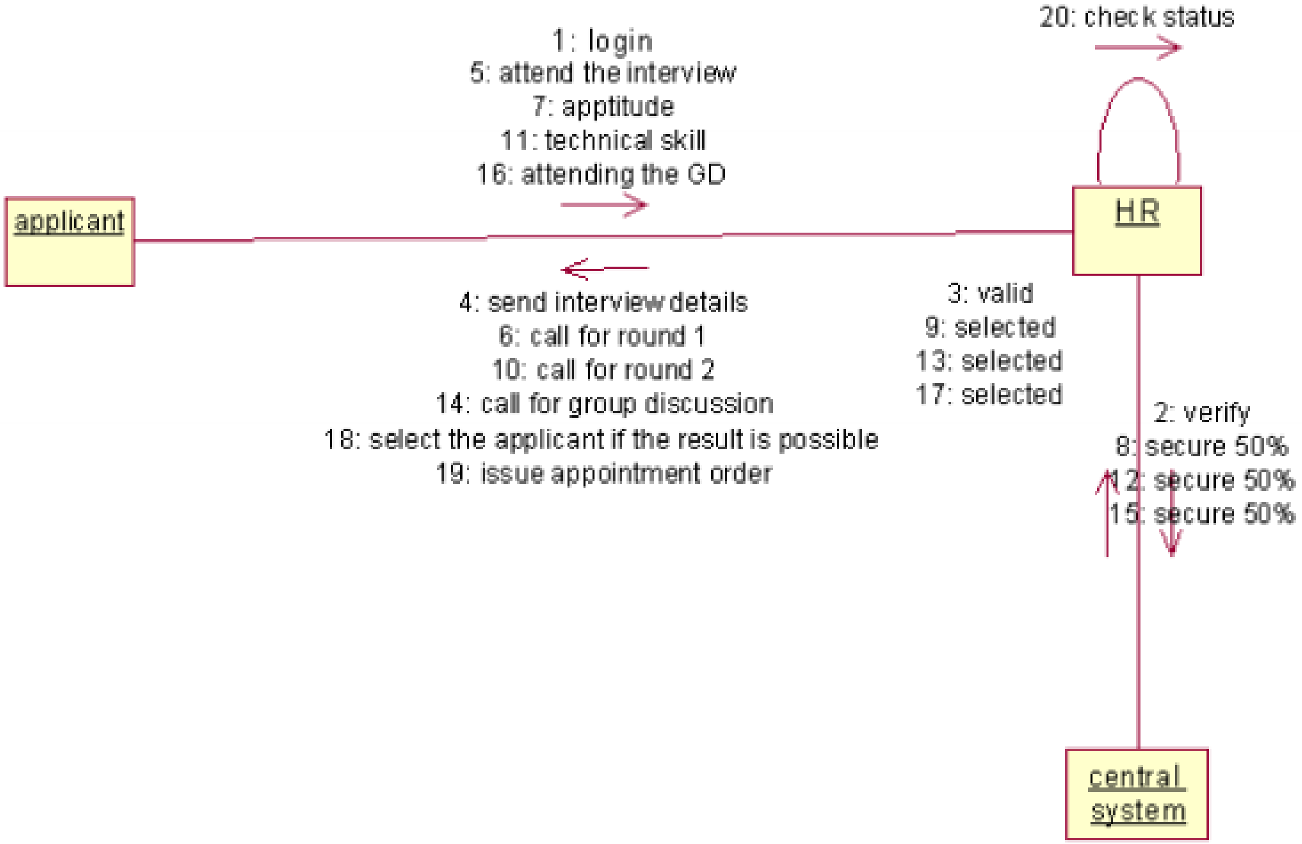
The Objects are

- Applicant
- HR
- Central System

The Messages are

1. Login
2. Verify
3. Valid
4. Send interview details
5. Attend the interview
6. Call for round 1
7. Aptitude
8. Secure 50%
9. Selected
10. Call for round 2
11. Technical skill
12. Secure 50%
13. Selected
14. Call for group discussion
15. Attending the GD
16. Secure 50%
17. Selected
18. Select the applicant if the result is positive
19. Issue appointment

Collaboration Diagram



COLLABORATION DIAGRAM

- It represents structural representation of objects.

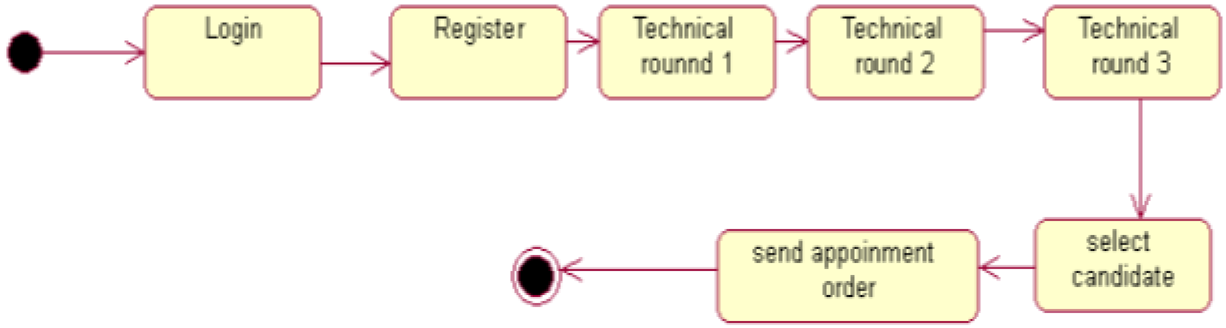
The Objects are

- Applicant
- HR
- Central system

The Messages are

1. Login
2. Verify
3. Valid
4. Send interview details
5. Attend the interview
6. Call for round 1
7. Aptitude
8. Secure 50%
9. Selected
10. Call for round 2
11. Technical skill
12. Secure 50%
13. Selected
14. Call for group discussion
15. Attending the GD
16. Secure 50%
17. Selected
18. Select the applicant if the result is positive
19. Issue appointment

Statechart Diagram



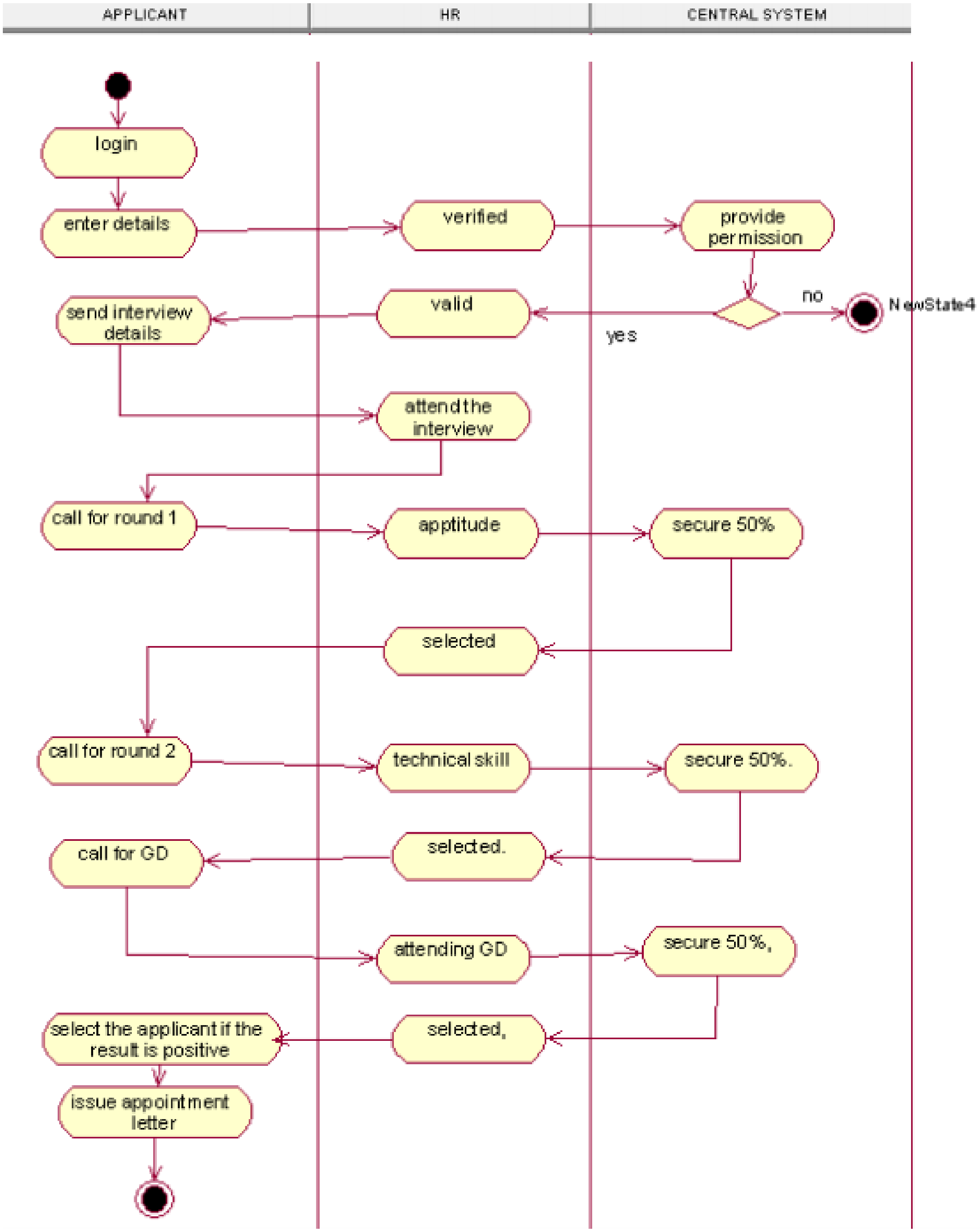
STATECHART DIAGRAM

- It depicts movement of one state to another when an event occurs. It consists of states and transitions.

The States are

- Login
- Register
- Technical round 1
- Technical round 2
- Technical round 3
- Select candidate
- Send appointment order

Activity Diagram



ACTIVITY DIAGRAM

- It depicts flow from one activity to other.

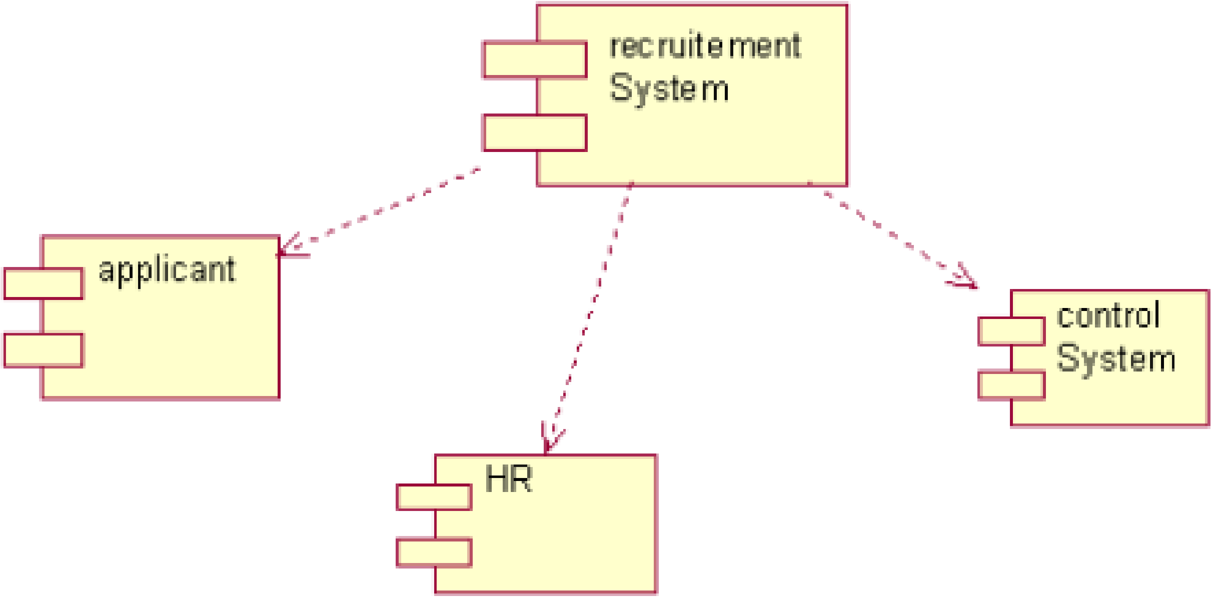
The Swimlanes are

- Applicant
- HR
- Central system

The Activities are

- Login
- Enter details
- Verified
- Provide permission
- Valid
- Send interview details
- Attend the interview
- Call for round 1
- Aptitude
- Secure 50%
- Selected
- Call for round 2
- Technical skill
- Secure 50%
- Selected
- Call for Group Discussion
- Attending Group Discussion
- Secure 50%
- Selected
- Select the applicant if the result is positive
- Issue appointment letter

Component Diagram



COMPONENT DIAGRAM

- It gives implementation details of the system.

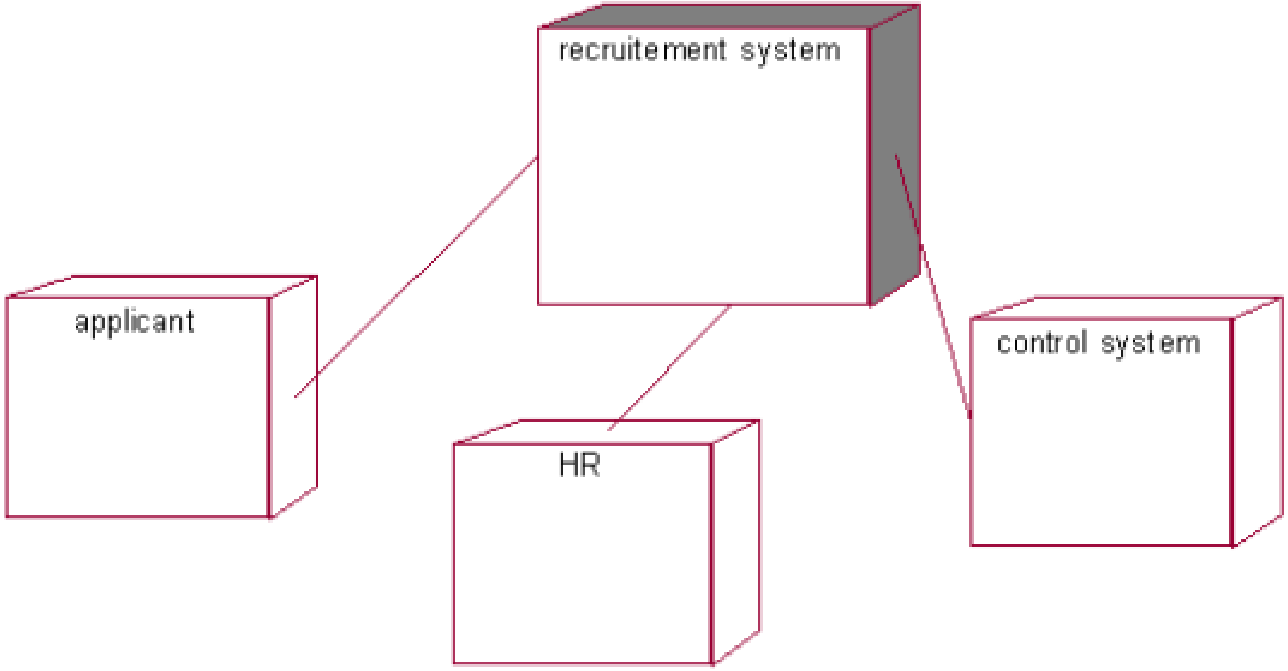
The components are

- Recruitment system
- Central system
- Applicant
- HR

The Relationships are

- Dependency

Deployment Diagram



DEPLOYMENT DIAGRAM

- It consists of nodes to deploy the application.

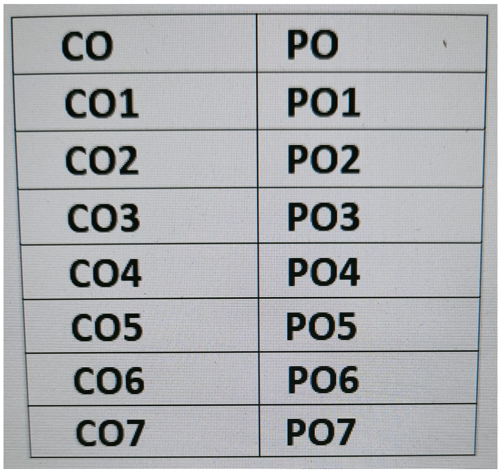
The Nodes are

- Recruitment system
- Applicant
- HR
- Central system

The Relationships are

- Association

Result : **Thus all the views of recruitment procedure for a software industry has been modelled and it is mapped to the following cos and pos.**



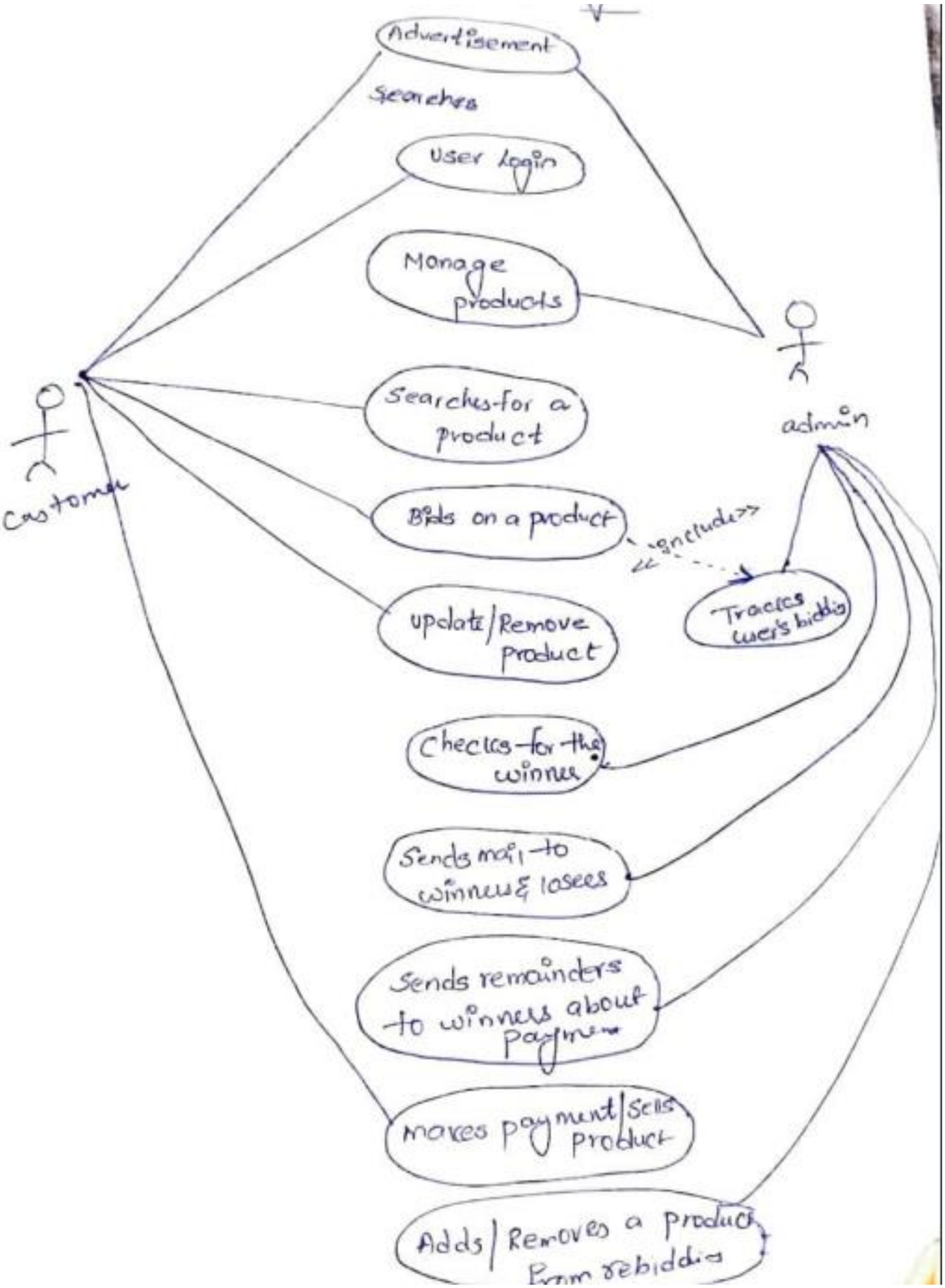
CO	PO
CO1	PO1
CO2	PO2
CO3	PO3
CO4	PO4
CO5	PO5
CO6	PO6
CO7	PO7

Aim : To model all the views of Online Auction Sales

Problem Statement :

The online auction system is a design about a website where sellers collect and prepare a list of items they want to sell and place it on the website for visualizing. To accomplish this purpose the user has to access the site. Incase it's a new user he has to register. Purchaser's login and select items they want to buy and keep bidding for it. Interacting with the purchasers and sellers through messages does this. There is no need for customer to interact with the sellers because every time the purchasers bid, the details will be updated in the database. The purchaser making the highest bid for an item before the close of the auction is declared as the owner of the item. If the auctioneer or the purchaser doesn't want to bid for the product then there is fixed cutoff price mentioned for every product. He can pay that amount directly and own the product. The purchaser gets a confirmation of his purchase as an acknowledgement from the website. After the transition by going back to the main menu where he can view other items.

Usecase Diagram



USECASE VIEW

- Uscase diagrams model the functionality of a system using actors and usecases which are a set of sequence of actions.

The Usecases are

- Advertisement
- User login
- Manage products
- Searches for a product
- Bids on a product
- Update/Remove product
- Checks fir the winner
- Sends mail to winner & losers
- Sends remainders to winner about payment
- Makes payment/ Sells product
- Adds/Removes a product from rebidding
- Tracks user's bidding

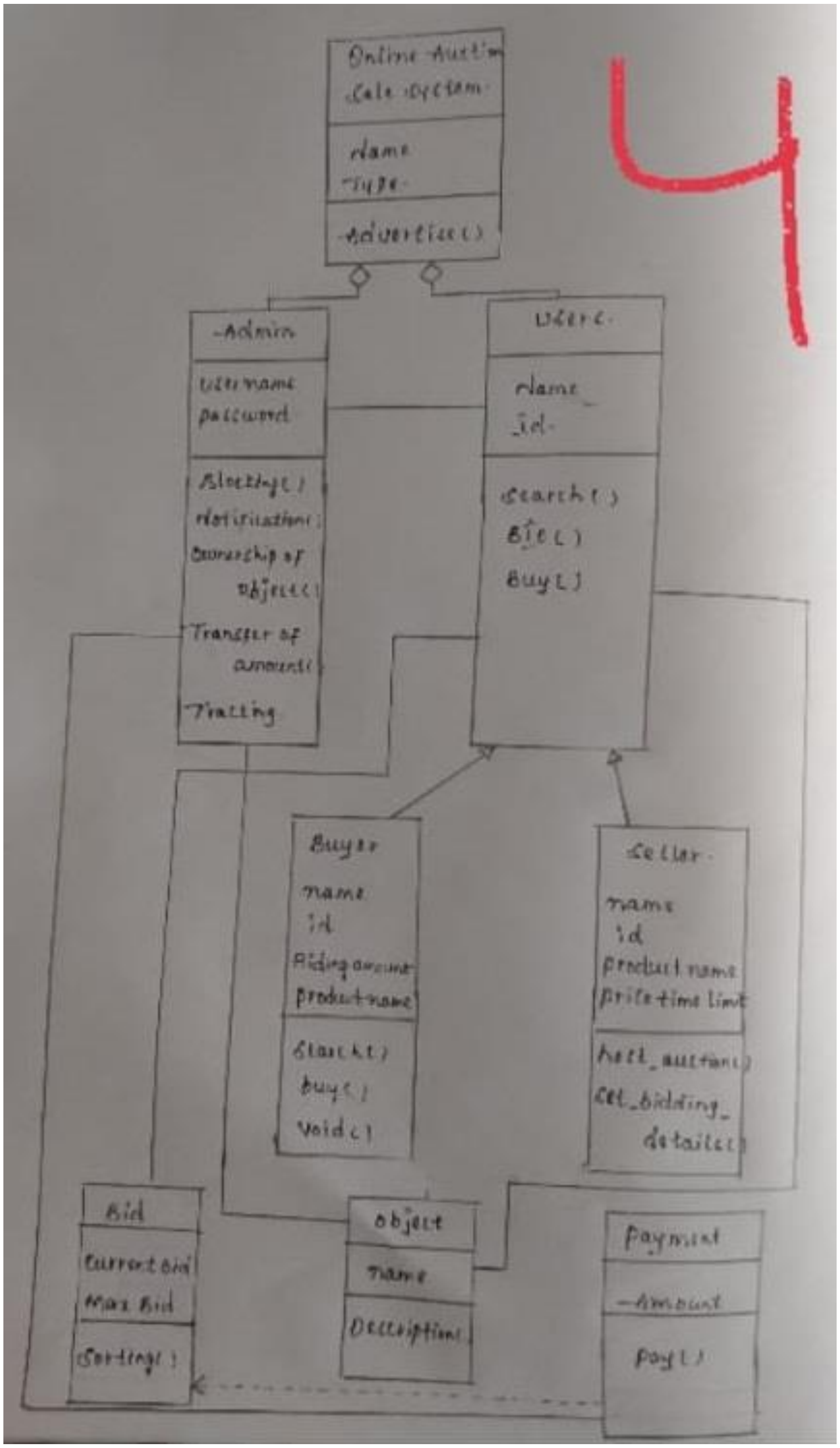
The Actors are

- Admin
- Customer

The Relationships are

- Association

Class Diagram



CLASS DIAGRAM

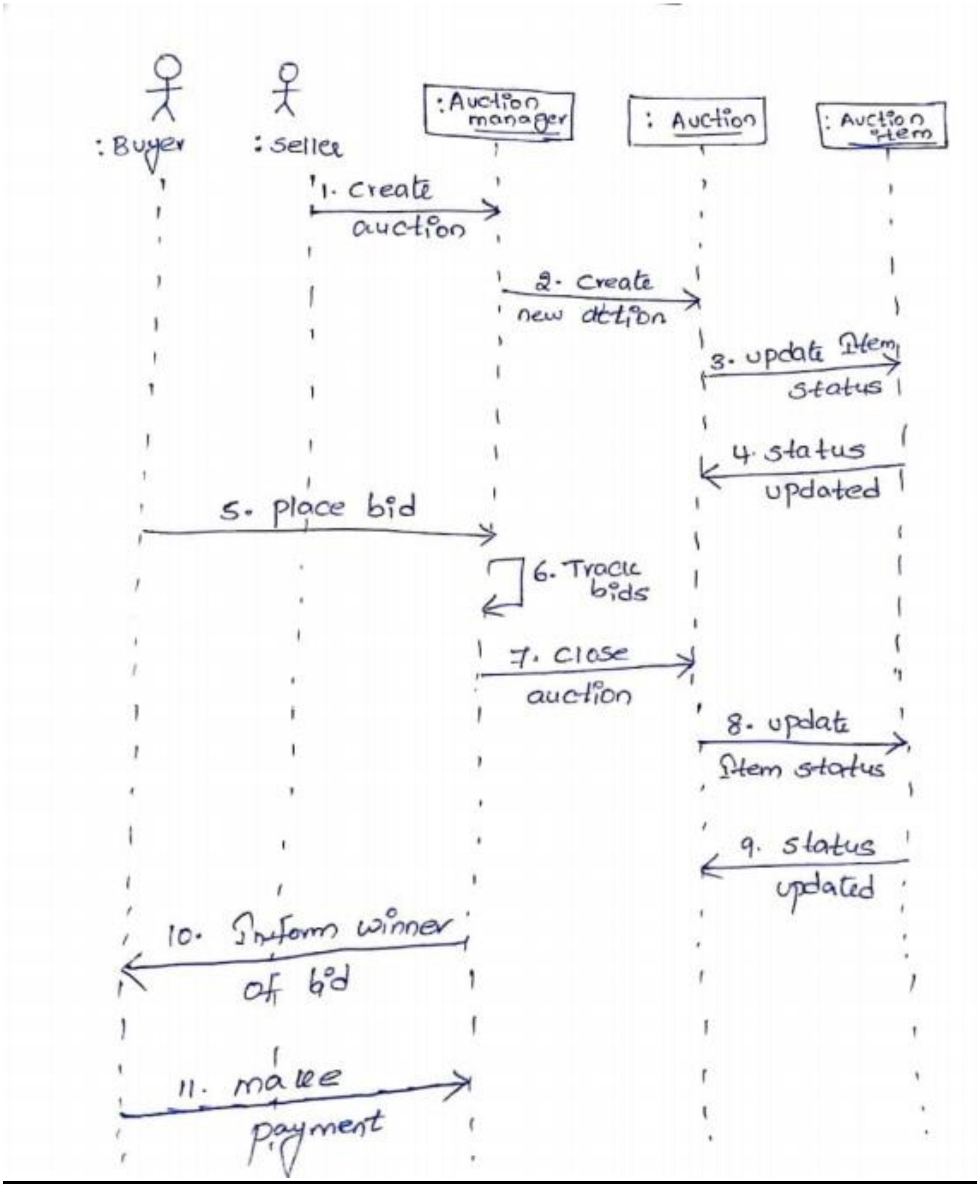
- Class diagram describes the structure of a system.

S.No.	Class Name	Attributes	Operations
1.	Online Auction Sale system	Name, Type	advertise()
2.	Admin	Username, Password	blocking(), notifications(), ownershipOfObject(), transferOfAmount(), tracking()
3.	Users	Name, ID	search(), bid(), buy()
4.	Buyer	Name, ID, Biding amount, Product name	search(), buy(), void()
5.	Seller	Name, ID, Product name, Price, Time limit	hostAuction(), setBidding(), details()
6.	Bid	Current bid, Max bid	sorting()
7.	Object	Name	description()
8.	Payment	Amount	pay()

The Relationships are

- Association
- Generalization
- Dependency

Sequence Diagram



SEQUENCE DIAGRAM

- It gives time ordering of messages.

The Objects are

- Auction manager
- Auction
- Auction item

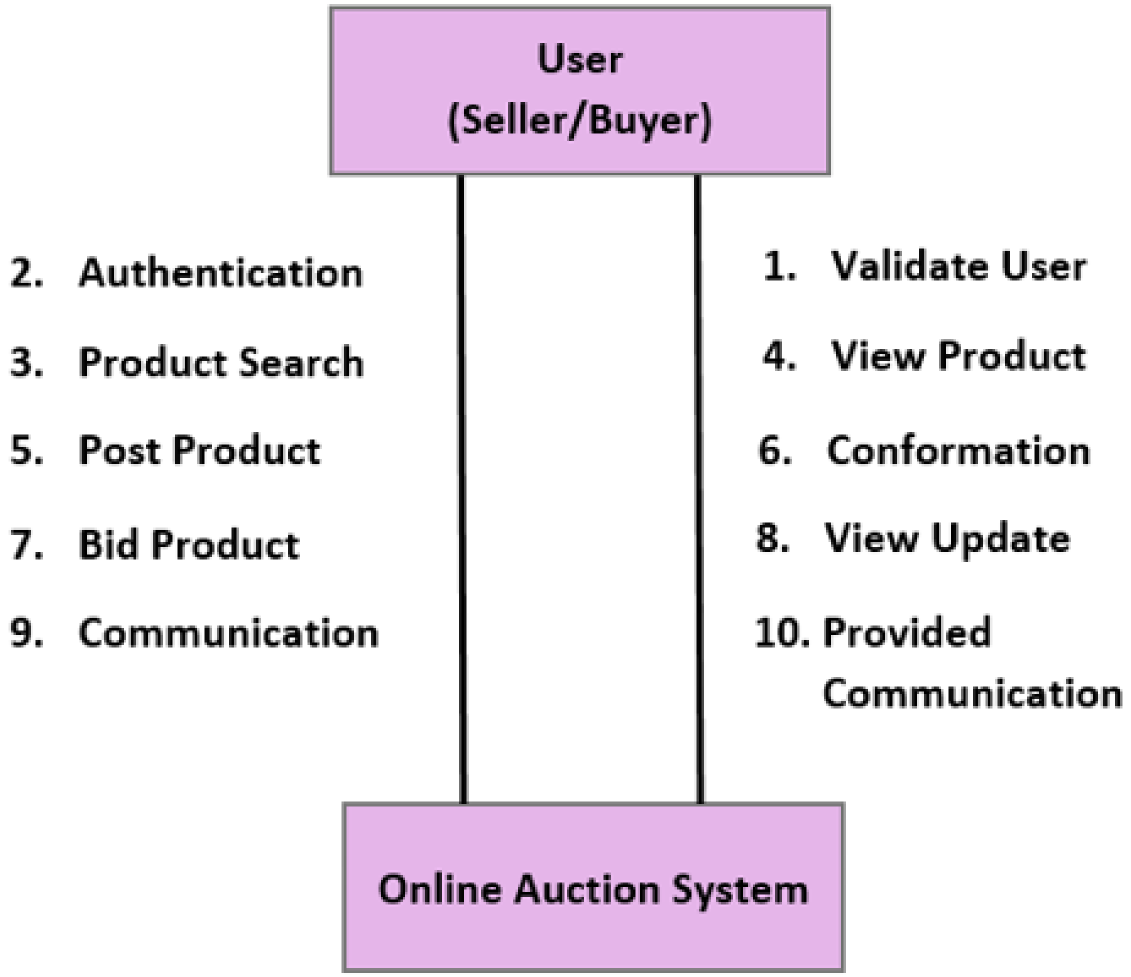
The Messages are

1. Create auction
2. Create new auction
3. Update item status
4. Status updated
5. Place bid
6. Track bids
7. Close auction
8. Update item status
9. Status updated
10. Inform winner of bid
11. Make payment

The Actors are

- Buyer
- Seller

Collaboration Diagram



COLLABORATION DIAGRAM

- It represents structural representation of objects.

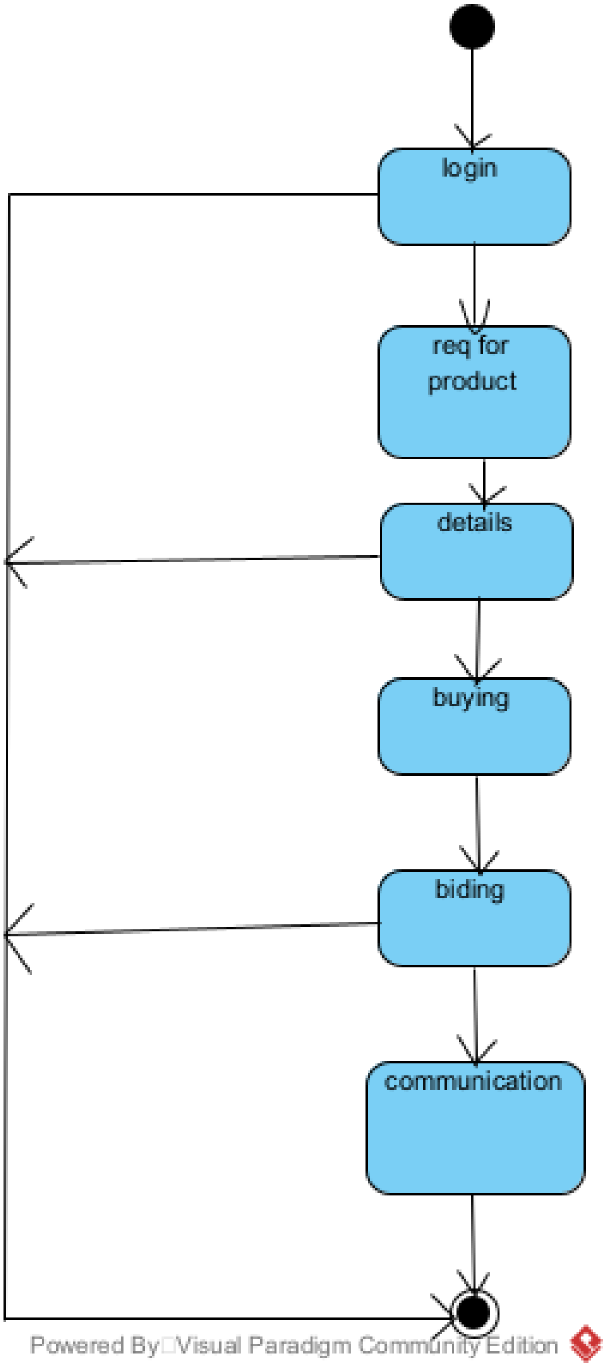
The Objects are

- User
- Online Auction system

The Messages are

1. Validate user
2. Authentication
3. Product search
4. View product
5. Post product
6. Confirmation
7. Bid product
8. View update
9. Communication
10. Provided communication

Statechart Diagram



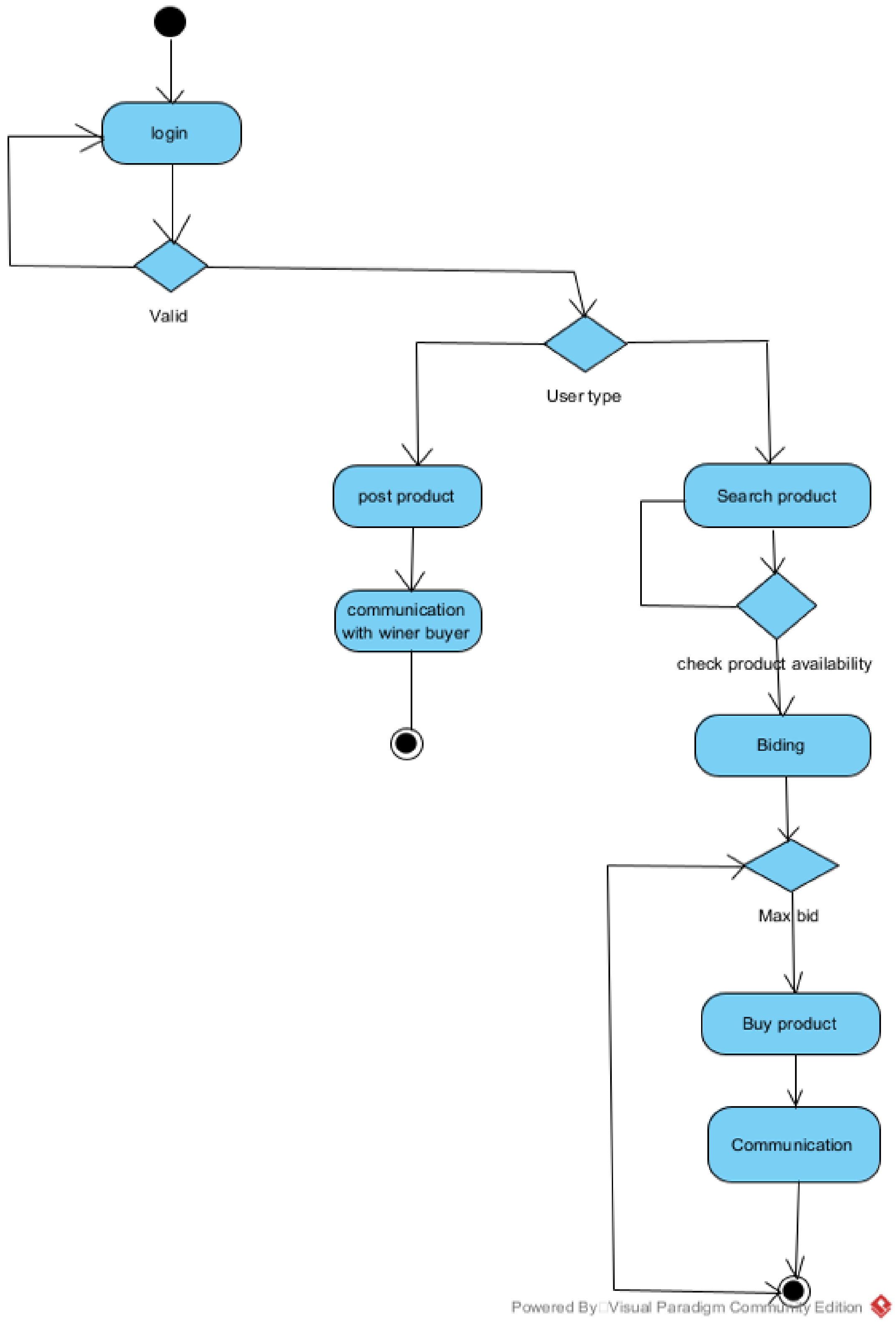
STATECHART DIAGRAM

- It depicts movement of one state to another when an event occurs. It consists of states and transitions.

The States are

- login
- request for product
- details
- buying
- biding
- communication

Activity Diagram


login
Valid
User type
post product
communication with winer buyer
Search product
check product availability
Biding
Max bid
Buy product
Communication
Powered By Visual Paradigm Community Edition

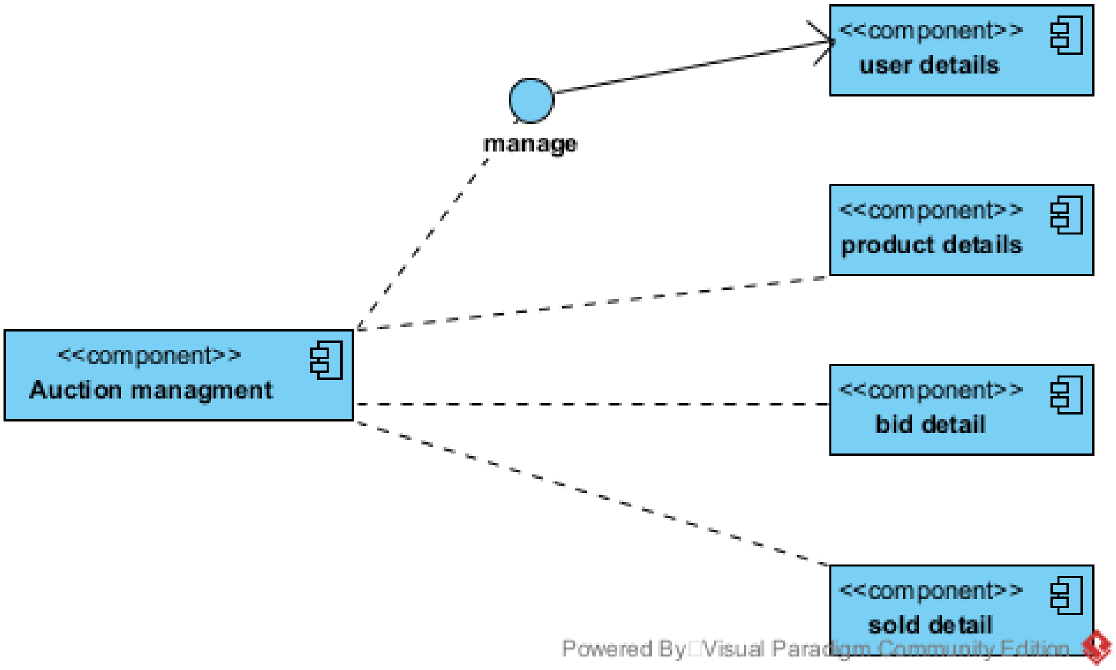
ACTIVITY DIAGRAM

- It depicts flow from one activity to other.

The Activities are

- Login
- Post product
- Communication with winner buyer
- Search product
- Biding
- Buy product
- Communication

Component Diagram



COMPONENT DIAGRAM

- It gives implementation details of the system.

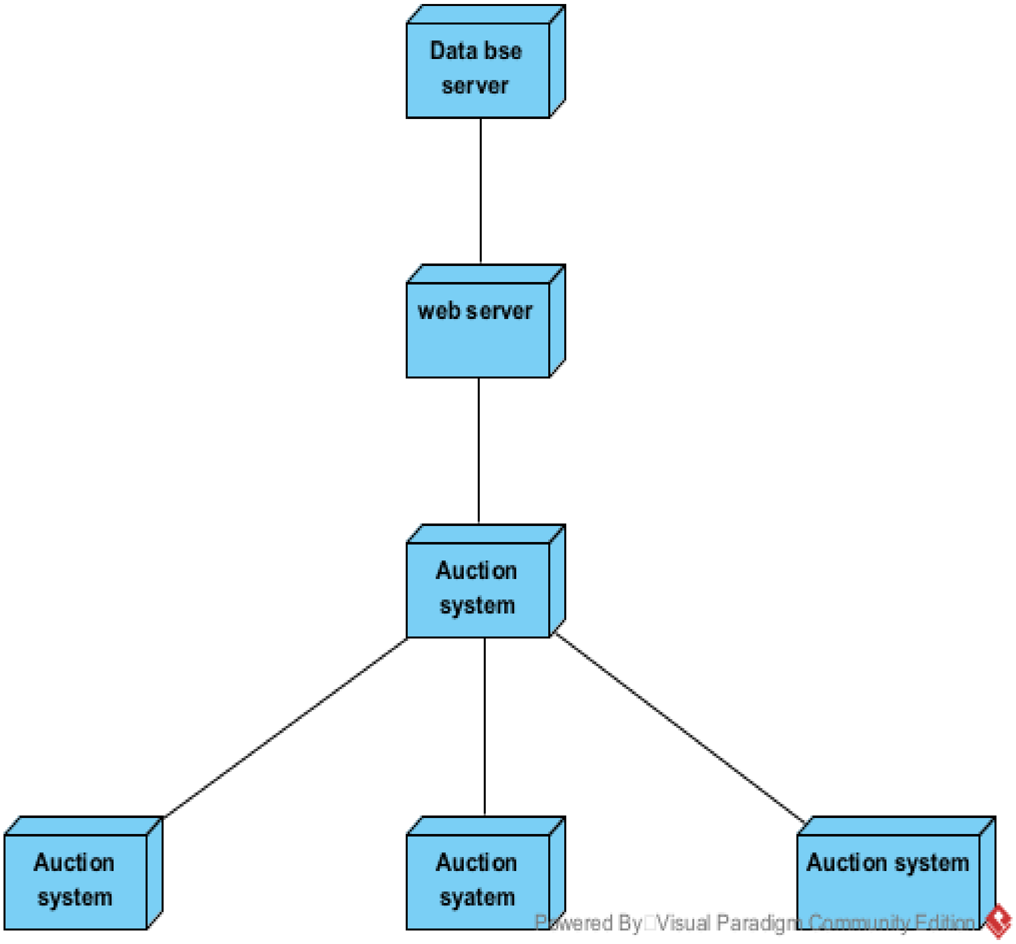
The Components are

- Auction management
- User details
- Product details
- Bid details
- Sold details

The Relationships are

- Association
- Dependency

Deployment Diagram



DEPLOYMENT DIAGRAM

- It consists of nodes to deploy the application.

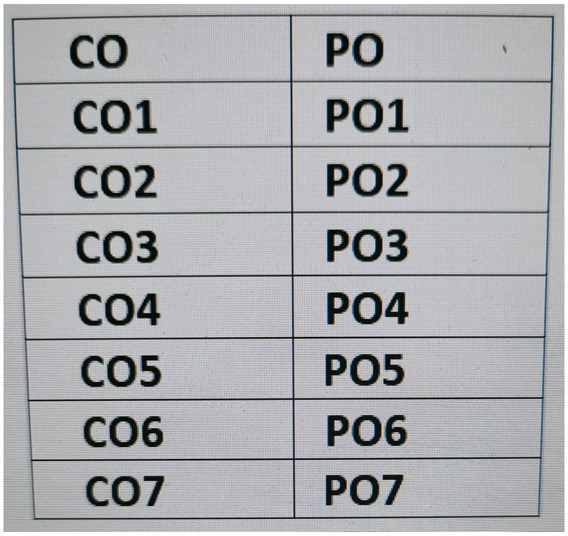
The Nodes are

- Database server
- Web server
- Auction system

The Relationships are

- Association

Result : **Thus all the views of online auction sales has been modelled and it has been mapped to following cos and pos.**



CO	PO
CO1	PO1
CO2	PO2
CO3	PO3
CO4	PO4
CO5	PO5
CO6	PO6
CO7	PO7

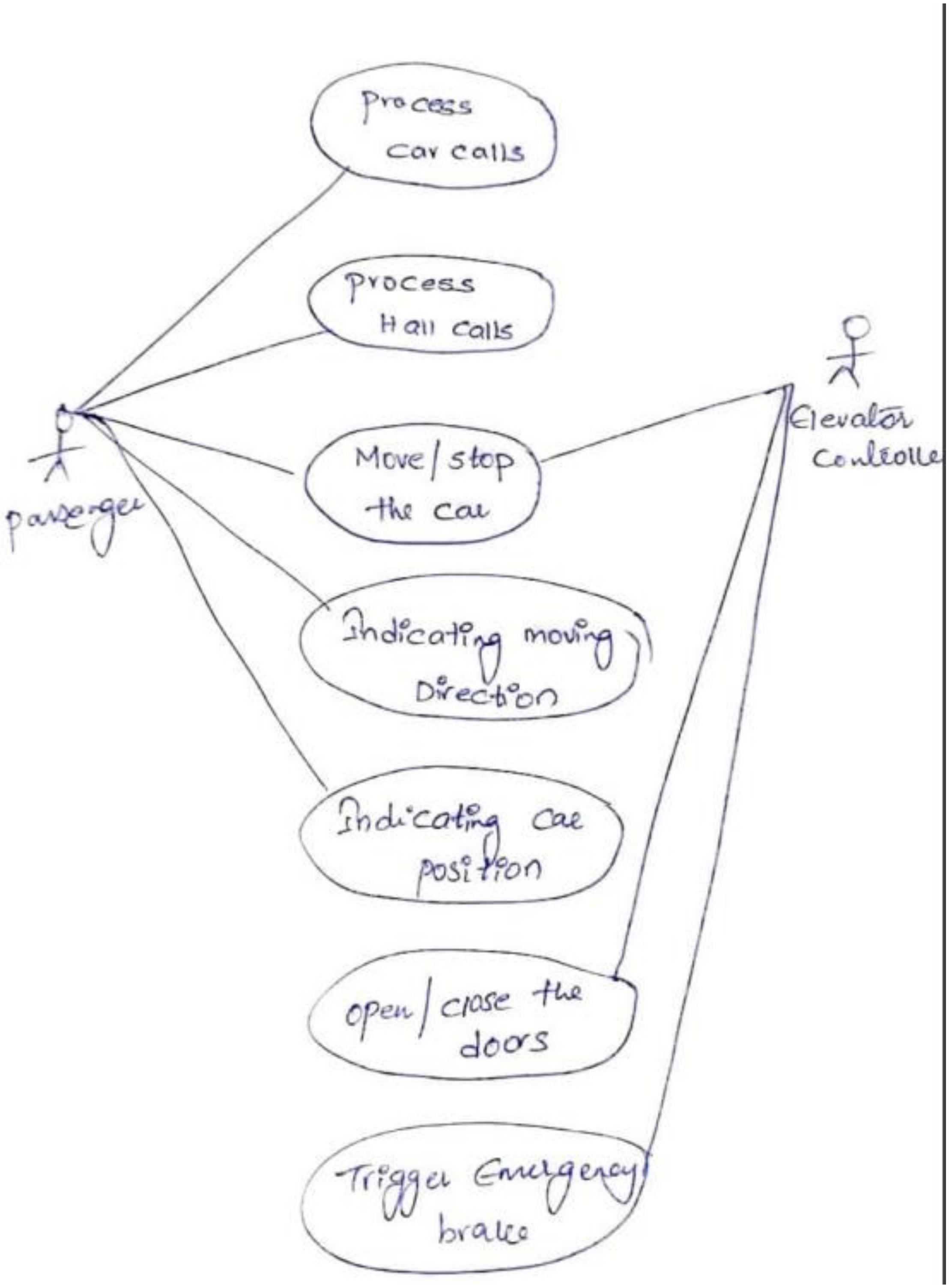
Aim : To model all the views of Two Floor Elevator Simulator

Problem Statement :

The elevator has the basic function that all elevator systems have, such as moving up and down, open and close doors, and of course, pick up passengers. The elevator is supposed to be used in a building having floors numbered from 1 to MaxFloor, where the first floor is the lobby. There are car call buttons in the car corresponding to each floor. For every floor except for the top floor and the lobby, there are two hall call buttons for the passengers to call for going up and down. There is only one down hall call button at the top floor and one up hall call button in the lobby. When the car stops at a floor, the doors are opened and the car lantern indicating the current direction the car is going is illuminated so that the passengers can get to know the current moving direction of the car. The car moves fast between floors, but it should be able to slow down early enough to stop at a desired floor. When an elevator has no requests, it remains at its current floor with its doors closed.

In order to certificate system safety, emergency brake will be triggered and the car will be forced to stop under any unsafe conditions.

<u>Usecase Diagram</u>



USECASE VIEW

- Uscase diagrams model the functionality of a system using actors and usecases which are a set of sequence of actions.

The Usecases are

- Process car calls
- Process hall calls
- Move/Stop the car
- Indicating moving direction
- Indicating car position
- Open/Close the doors
- Trigger emergency brakes

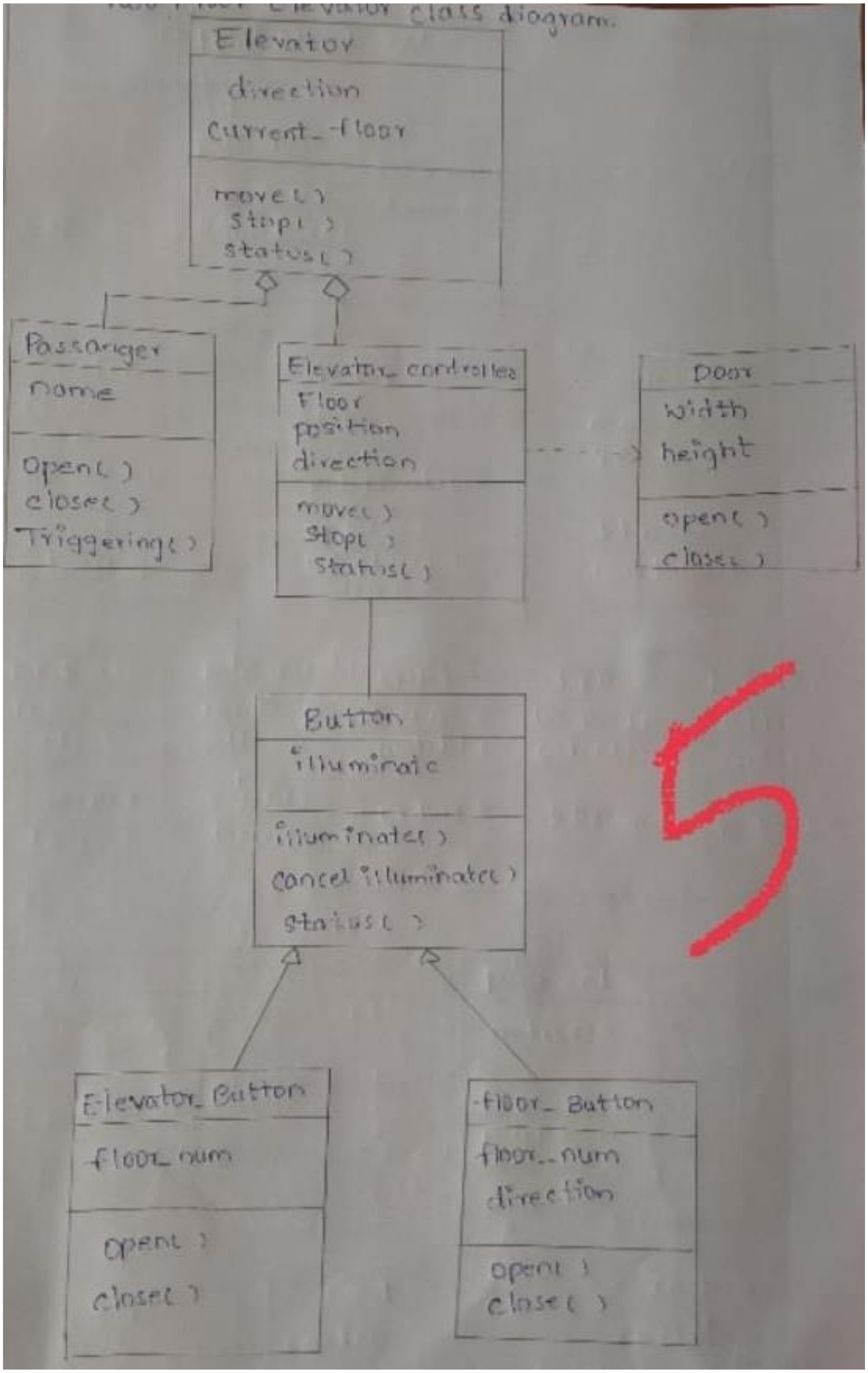
The Actors are

- Passenger
- Elevator controller

The Relationships are

- Association

<u>Class Diagram</u>



CLASS DIAGRAM

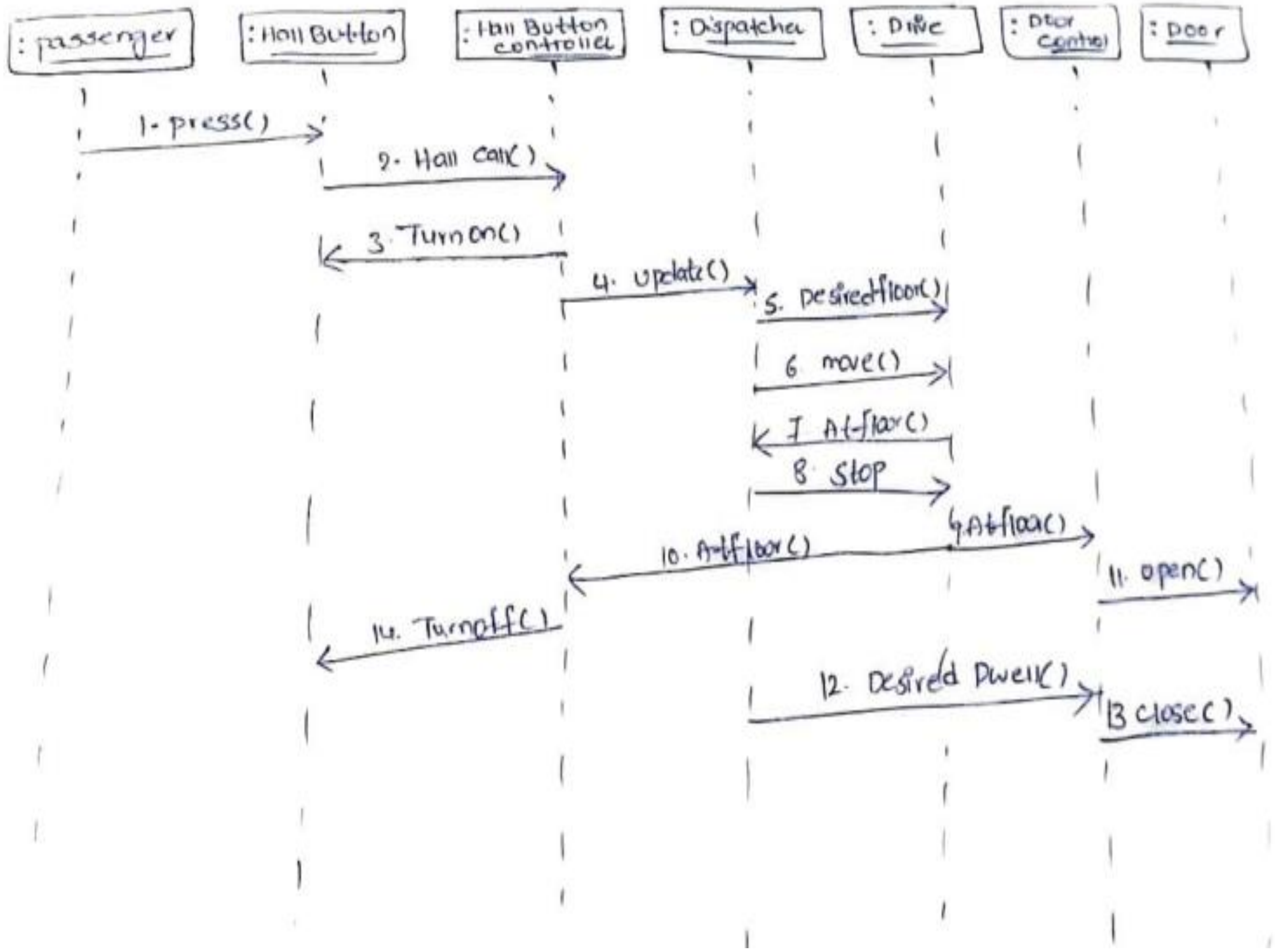
- Class diagram describes the structure of a system.

S.No	Class Name	Attributes	Operations
1.	Elevator	Direction, Current floor	move(), stop(), status()
2.	Passenger	Name	open(), close(), triggering()
3.	Elevator Controller	Floor, Position, Direction	move(), stop(), status()
4.	Door	Width, Height	open(), close()
5.	Button	Illuminate	illuminate(), cancelIllumination(), status()
6.	Elevator button	Floor number	open(), close()
7.	Floor button	Floor number, Direction	open(), close()

The Relationships are

- Association
- Dependency
- Generalization

Sequence Diagram



SEQUENCE DIAGRAM

- It gives time ordering of messages.

The Objects are

- Passenger
- Hall Button
- Hall Button Controller
- Dispatcher
- Drive
- Door Control
- Door

The Messages are

1. Press
2. Hall call
3. Turn on
4. Update
5. Desired floor
6. Move
7. At floor
8. Stop
9. At floor

10.At floor

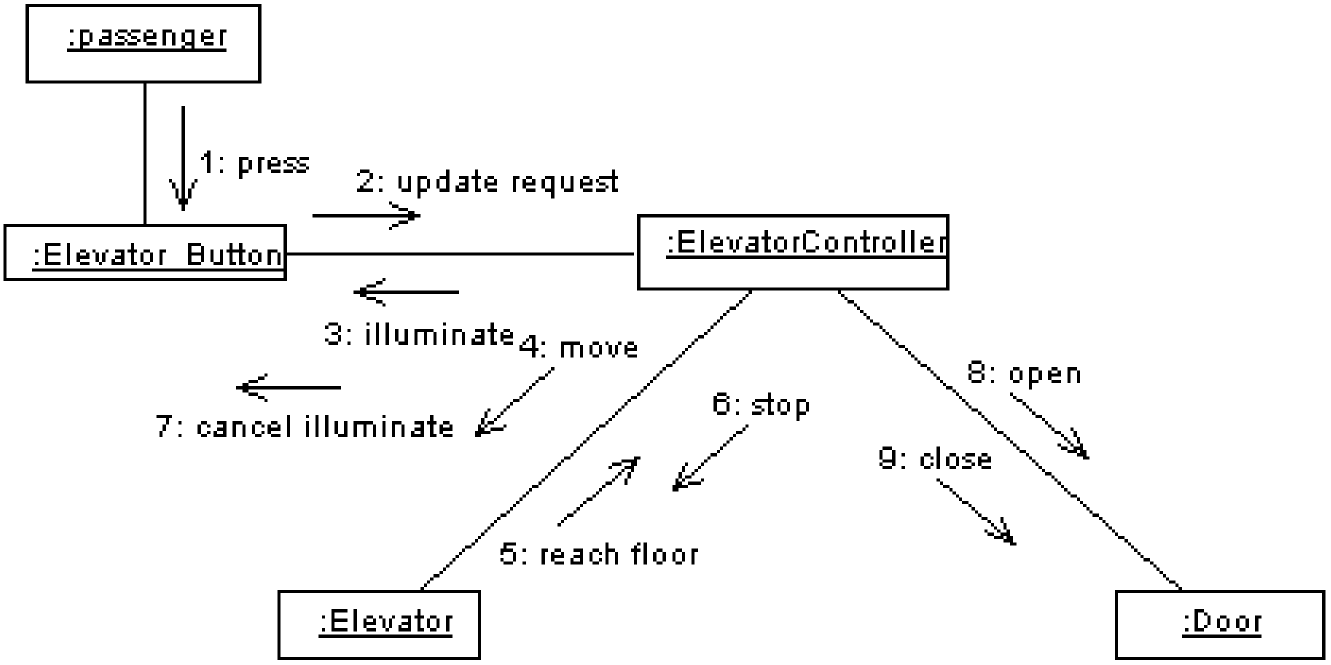
11.Open

12.Desired dwell

13.Close

14.Turnoff

Collaboration Diagram



COLLABORATION DIAGRAM

- It represents structural representation of objects.

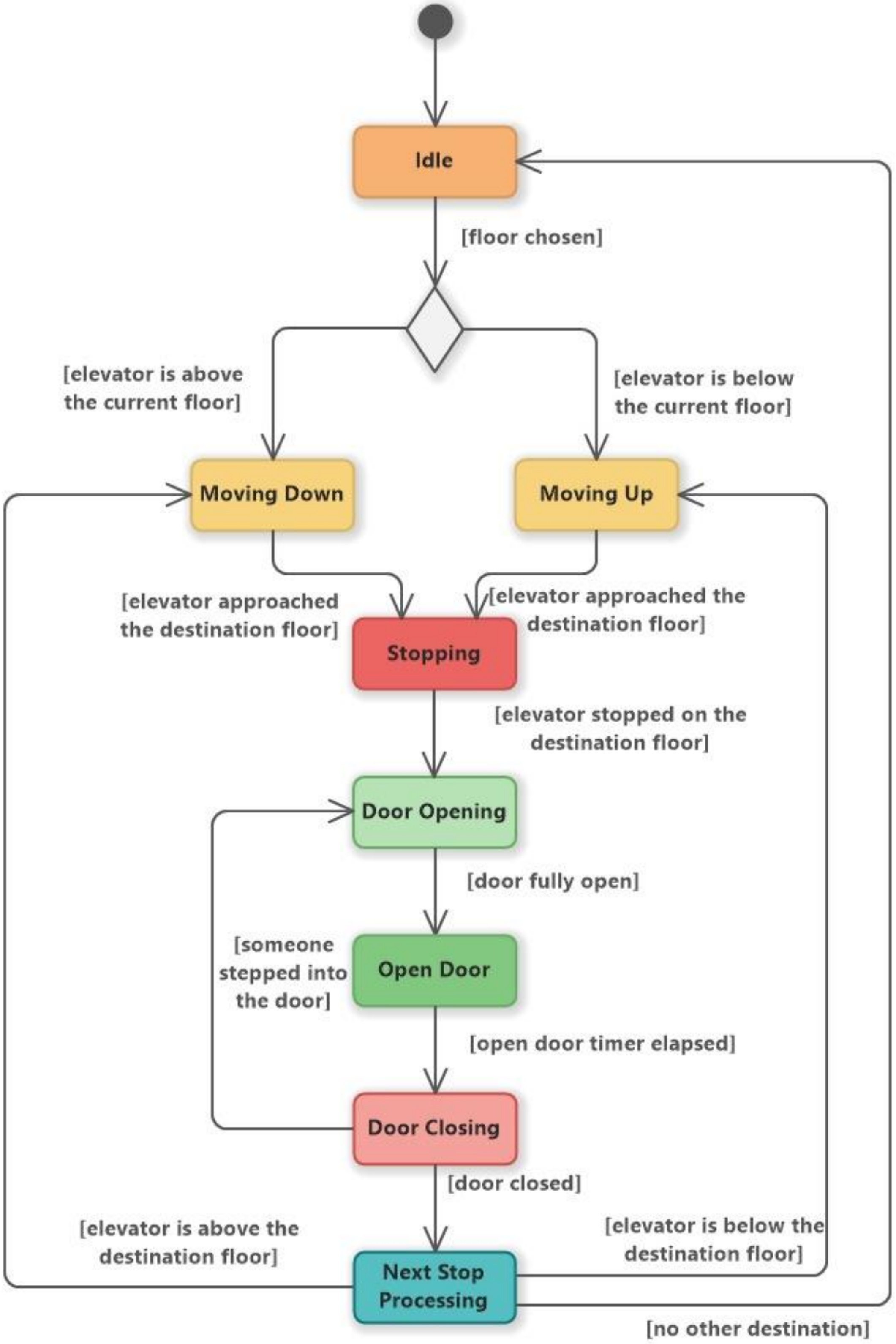
The Objects are

- Passenger
- Elevator button
- Elevator controller
- Elevator
- Door

The Messages are

1. Press
2. Update request
3. Illuminate
4. Move
5. Reach floor
6. Stop
7. Cancel illumination
8. Open
9. Close

Statechart Diagram


Idle
[floor chosen]
[elevator is above the current floor]
[elevator is below the current floor]
Moving Down
Moving Up
[elevator approached the destination floor]
[elevator approached the destination floor]
Stopping
[elevator stopped on the destination floor]
Door Opening
[door fully open]
[someone stepped into the door]
Open Door
[open door timer elapsed]
Door Closing
[door closed]
[elevator is above the destination floor]
[elevator is below the destination floor]
Next Stop Processing
[no other destination]

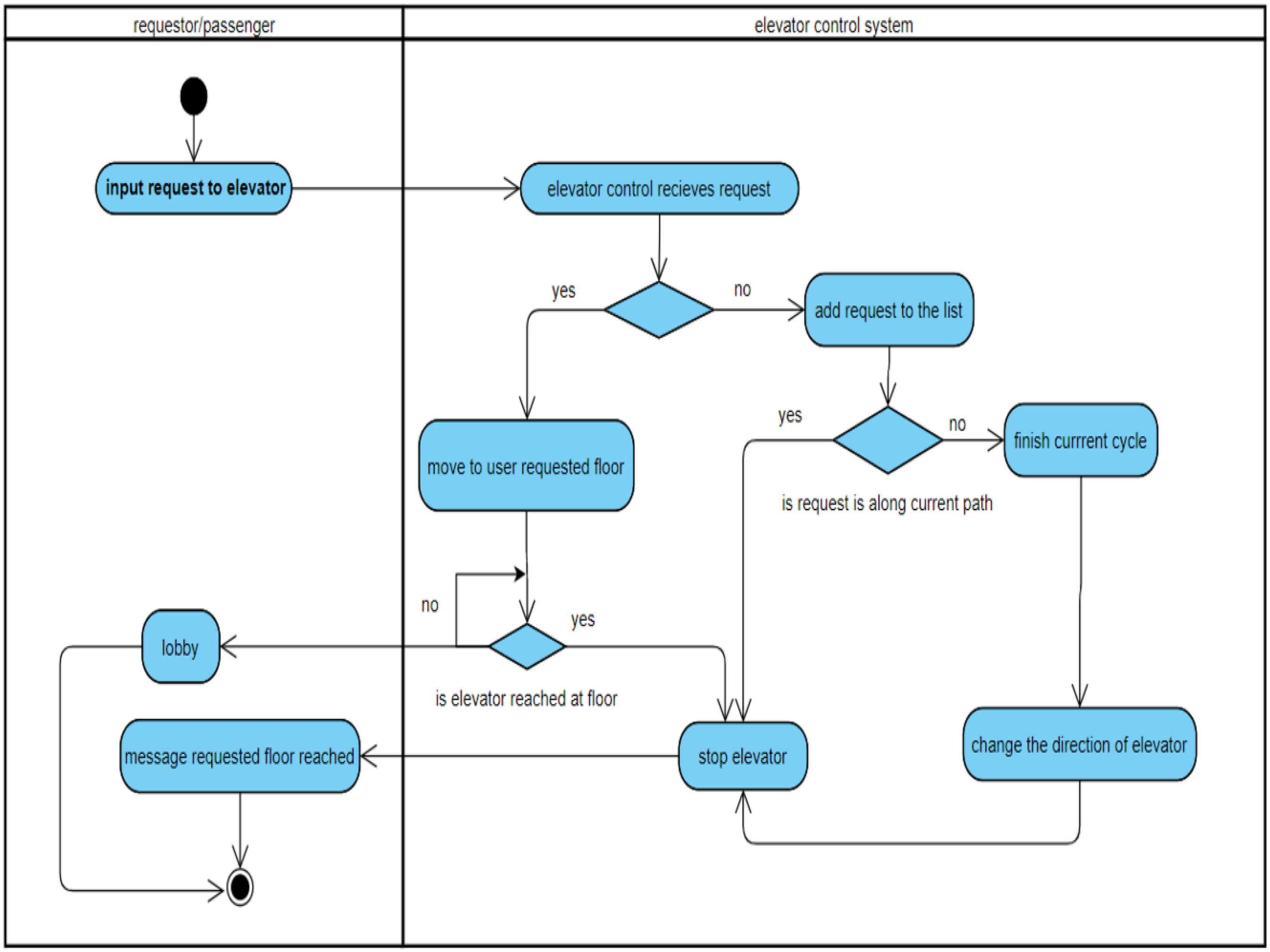
STATECHART DIAGRAM

- It depicts movement of one state to another when an event occurs. It consists of states and transitions.

The States are

- Idle
- Moving down
- Moving up
- Stopping
- Door opening
- Open door
- Door closing
- Next stop processing

Activity Diagram


requestor/passenger
elevator control system
input request to elevator
elevator control recieves request
yes
no
add request to the list
move to user requested floor
yes
no
finish currrent cycle
is request is along current path
no
yes
lobby
is elevator reached at floor
message requested floor reached
stop elevator
change the direction of elevator

ACTIVITY DIAGRAM

- It depicts flow from one activity to other.

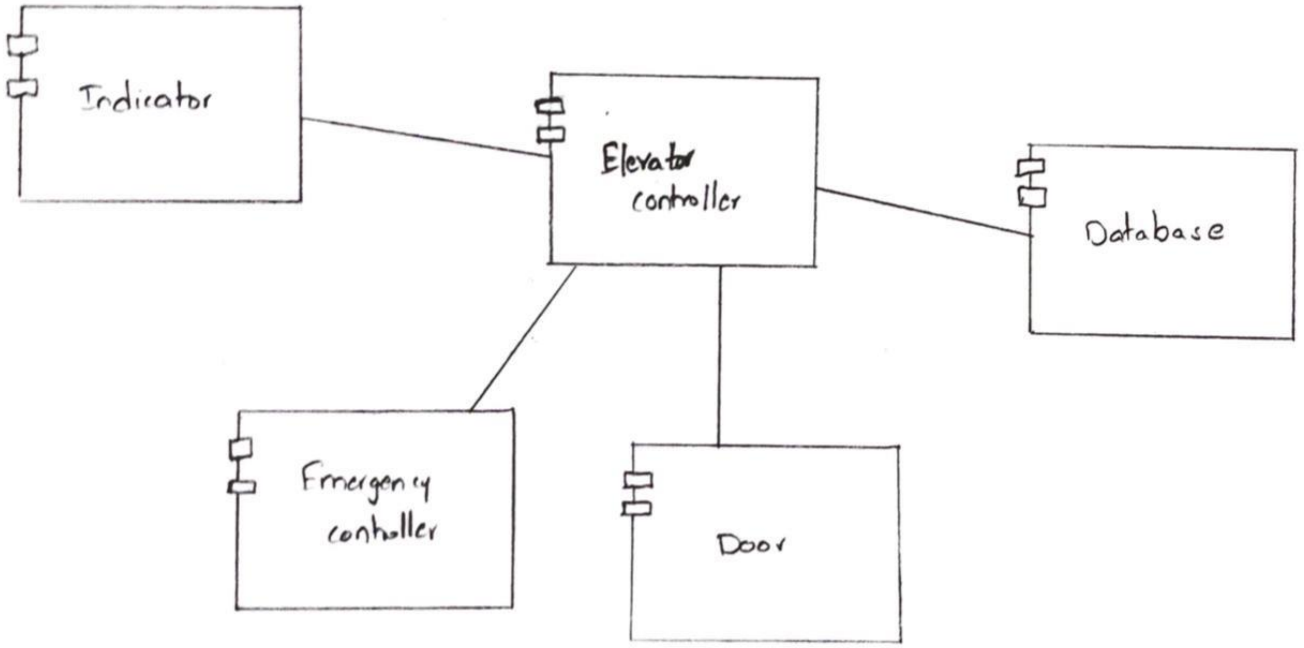
The Swimlanes are

- Requester/Passenger
- Elevator control system

The Activities are

- Input request to elevator
- Elevator control receives request
- Add request to the list
- Move the user to requested floor
- Finish current cycle
- Change the direction of elevator
- Stop elevator
- Lobby
- Message requested floor reached

Component Diagram



COMPONENT DIAGRAM

- It gives implementation details of the system.

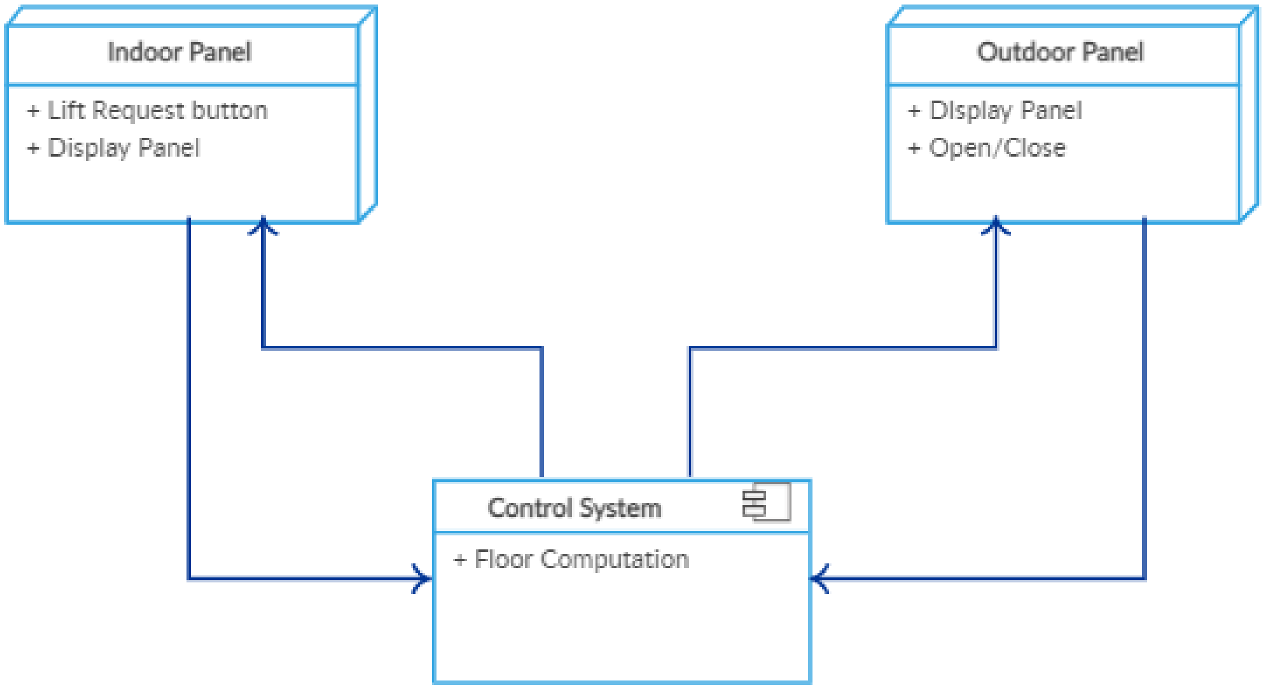
The Components are

- Elevator controller
- Databse
- Door
- Emergency controller
- Indicator

The Relationships are

- Association

<u>Deployment Diagram</u>



DEPLOYMENT DIAGRAM

- It consists of nodes to deploy the application.

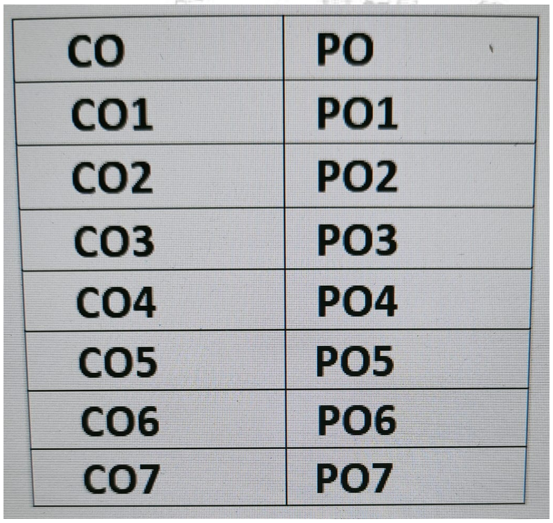
The Nodes are

- Indoor panel
- Outdoor panel
- Control system

The Relationships are

- Dependency

Result : Thus all the views of two floor elevator simulator has been modelled and it has been mapped to following cos and pos.



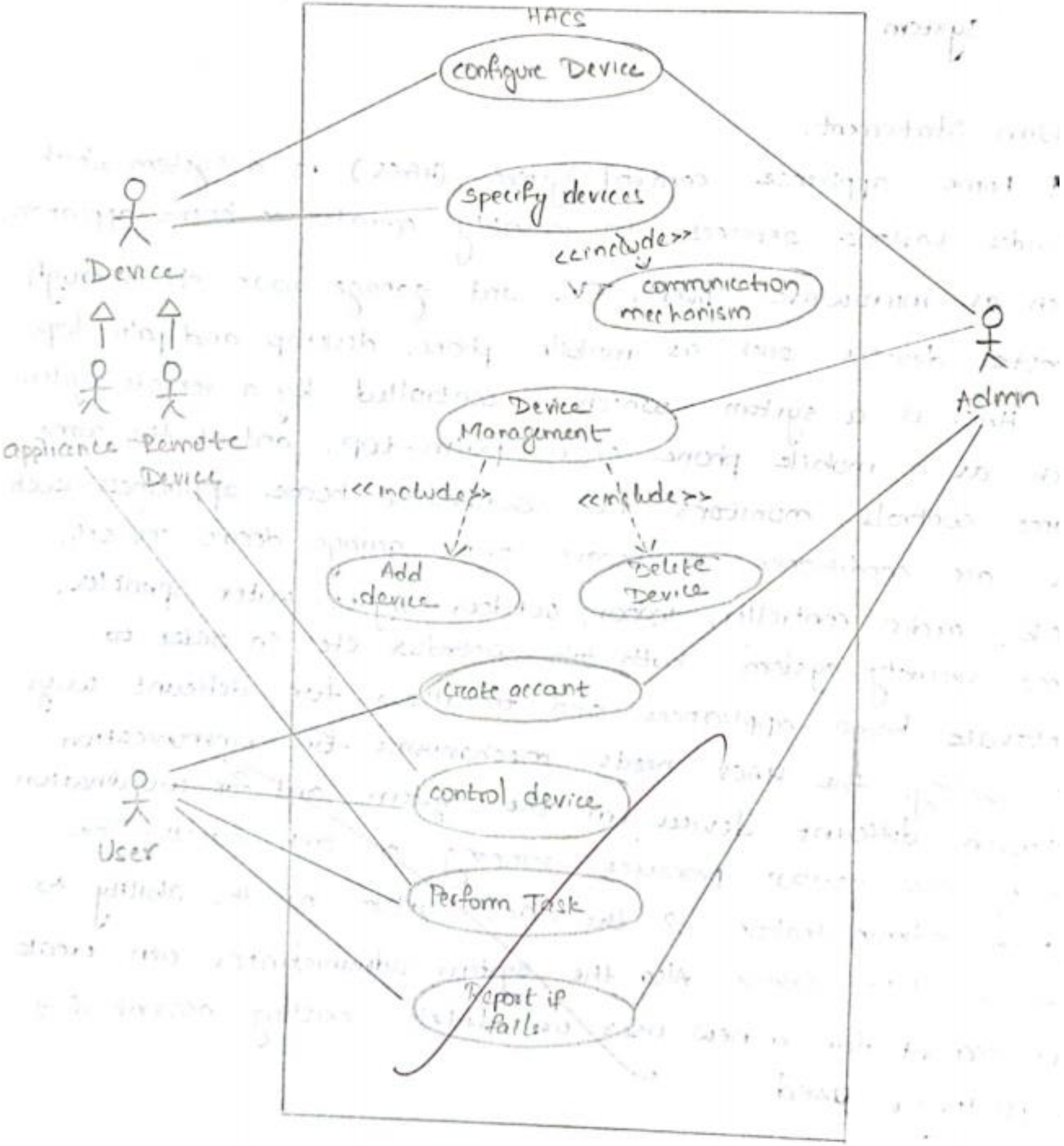
CO	PO
CO1	PO1
CO2	PO2
CO3	PO3
CO4	PO4
CO5	PO5
CO6	PO6
CO7	PO7

Aim : To model all the views of Home Appliance Control System (HACS)

Problem Statement :

A home appliance control system (HACS) is a system which provides various services to remotely operate on home appliances, such as microwave oven, TV, and garage door etc through remote devices such as mobile phone, desktop and palm-top.A home appliance control system (HACS) is a system which is controlled by a remote system such as a mobile phone or a palm-top, and at the same time controls, monitors and coordinates home appliances such as air conditioner, microwave oven, garage doors, TV set, VCR, audio controller, indoor/outdoor lights, water sprinkler, home security system, bath tub controller, etc. In order to activate home appliances and to allow for different ways of cooking, the HACS needs mechanisms for communication between the different devices in the system, and for coordination among the various processes running on such devices. The system administrator of the HACS system has the ability to add a new appliance or delete an existing one. The system administrator has the ability to add a new remote device and configure it with HACS or delete an existing one when it is not used. Also the system administrator can create an account for a new user or delete existing account if it is no longer used.

Usecase Diagram



USECASE VIEW

- Uscase diagrams model the functionality of a system using actors and usecases which are a set of sequence of actions.

The Actors are

- Admin
- Device
- Appliance
- Remote Device
- User

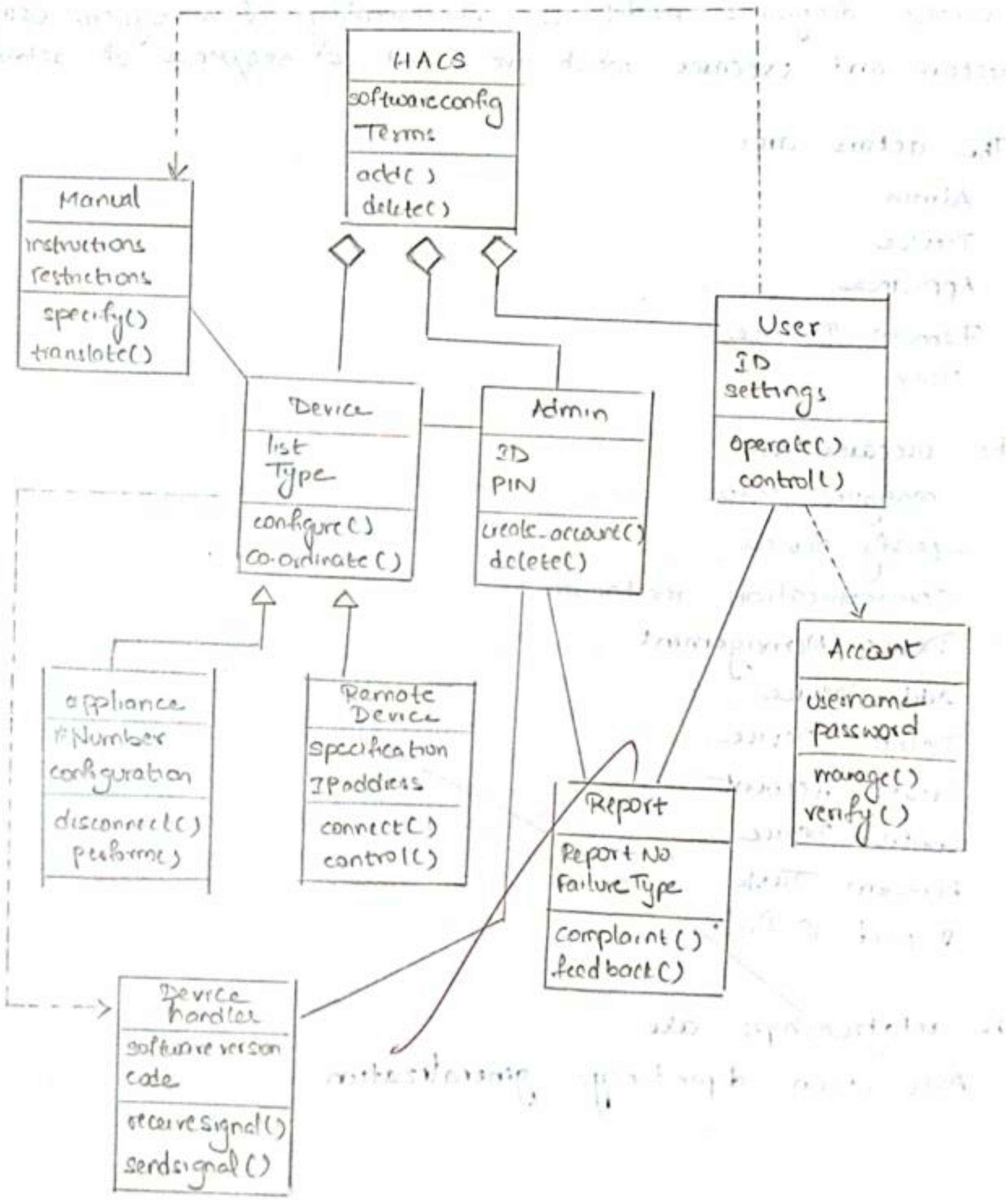
The Usecases are

- Configure device
- Specify devices
- Communication mechanism
- Device management
- Add device
- Delete device
- Create device
- Control device
- Perform device
- Report if fails

The Relationships are

- Association
- Dependency
- Generalization

Class Diagram



CLASS DIAGRAM

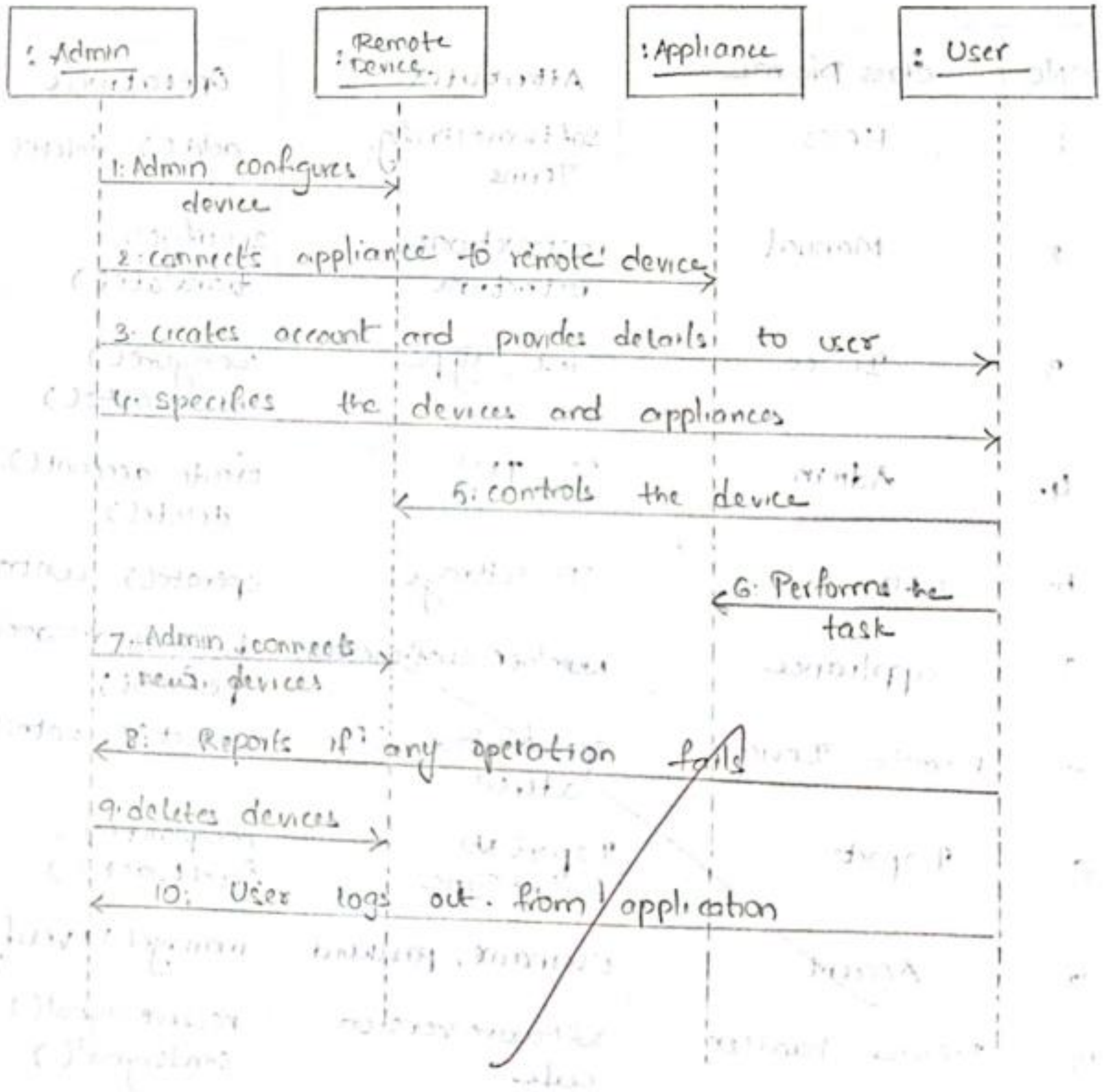
- Class diagram describes the structure of a system.

S.No	Class Name	Attributes	Operations
1.	HACS	SoftwareConfigTerms	add(), delete()
2.	Manual	Instructions, Restrictions	specify(), translate()
3.	Device	List, Type	configure(), coordinate()
4.	Admin	ID, Pin	createAccount(), delete()
5.	User	ID, Settings	operate(), control()
6.	Appliance	Number, Configuration	disconnect(), perform()
7.	Remote device	Specification, IP Address	connect(), control()
8.	Report	Report No., Failure type	complaint(), feedback()
9.	Account	Username, Password	manage(), verify()
10.	Device handler	Software version, Code	receiveSignal(), sendSignal()

The Relationships are

- Association
- Dependency
- Generalization

Sequence Diagram


: Admin
: Remote Device.
: Appliance
: User
1: Admin configures device
2. connects appliance to remote device
3. creates account and provides details to user
4. specifies the devices and appliances
5. controls the device
6. Performs the task
7. Admin connects new devices
8: Reports if any operation fails
9. deletes devices
10: User logs out from application

SEQUENCE DIAGRAM

- It gives time ordering of messages.

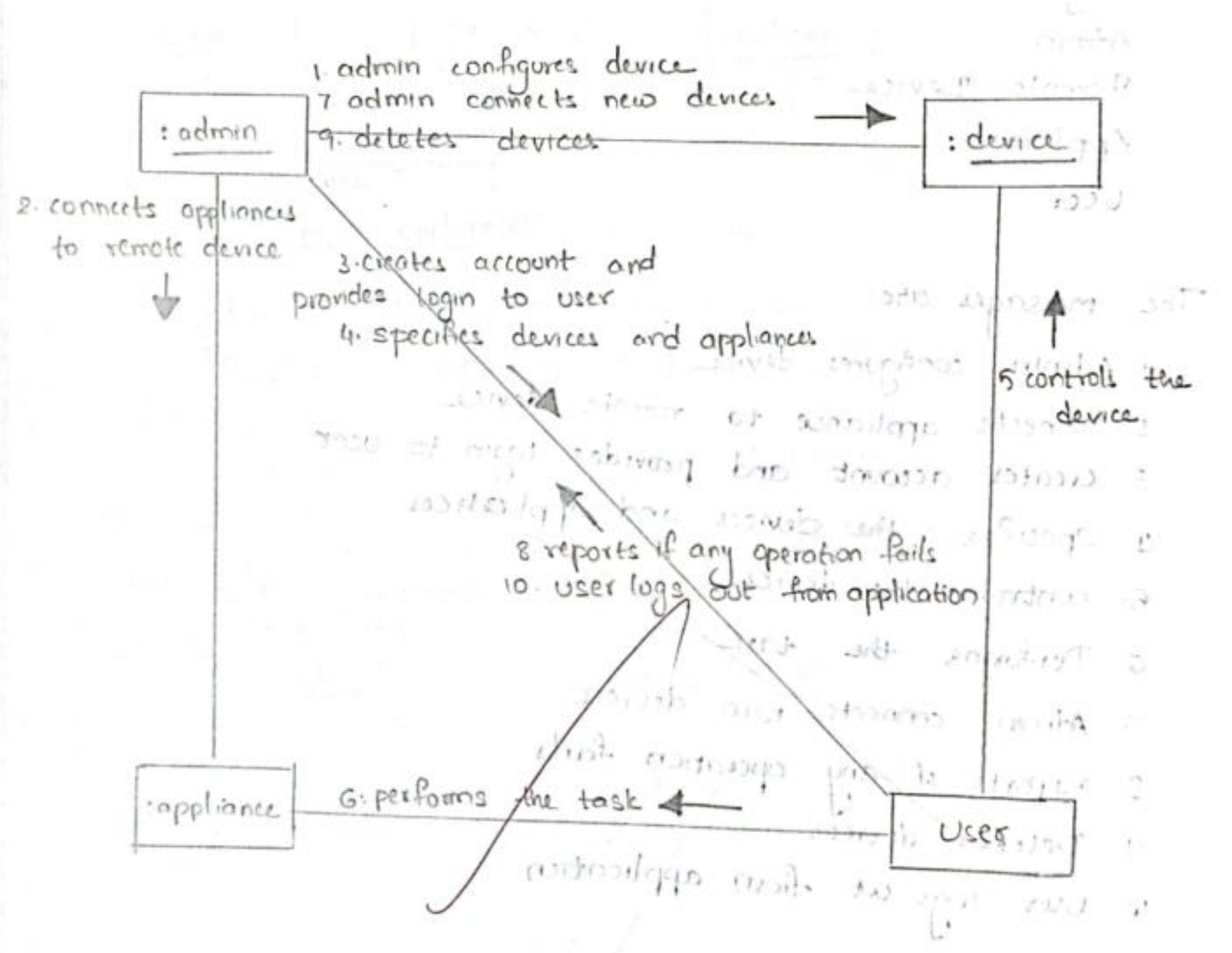
The Objects are

- Admin
- Remote device
- Appliance
- User

The Messages are

1. Admin configures device
2. Connects appliance to remote device
3. Creates account and provides login to user
4. Specifies the devices and appliances
5. Controls the device
6. Performs the task
7. Admin connects new devices
8. Report if any operation fails
9. Deletes devices
10. User logs out from application

<u>Collaboration Diagram</u>


: admin
: device
:appliance
User
1. admin configures device
7 admin connects new devices
9. deletes devices
2. connects appliances to remote device
3. creates account and provides login to user
4. specifies devices and appliances
5 controls the device
6: performs the task
8 reports if any operation fails
10. user logs out from application

COLLABORATION DIAGRAM

- It represents structural representation of objects.

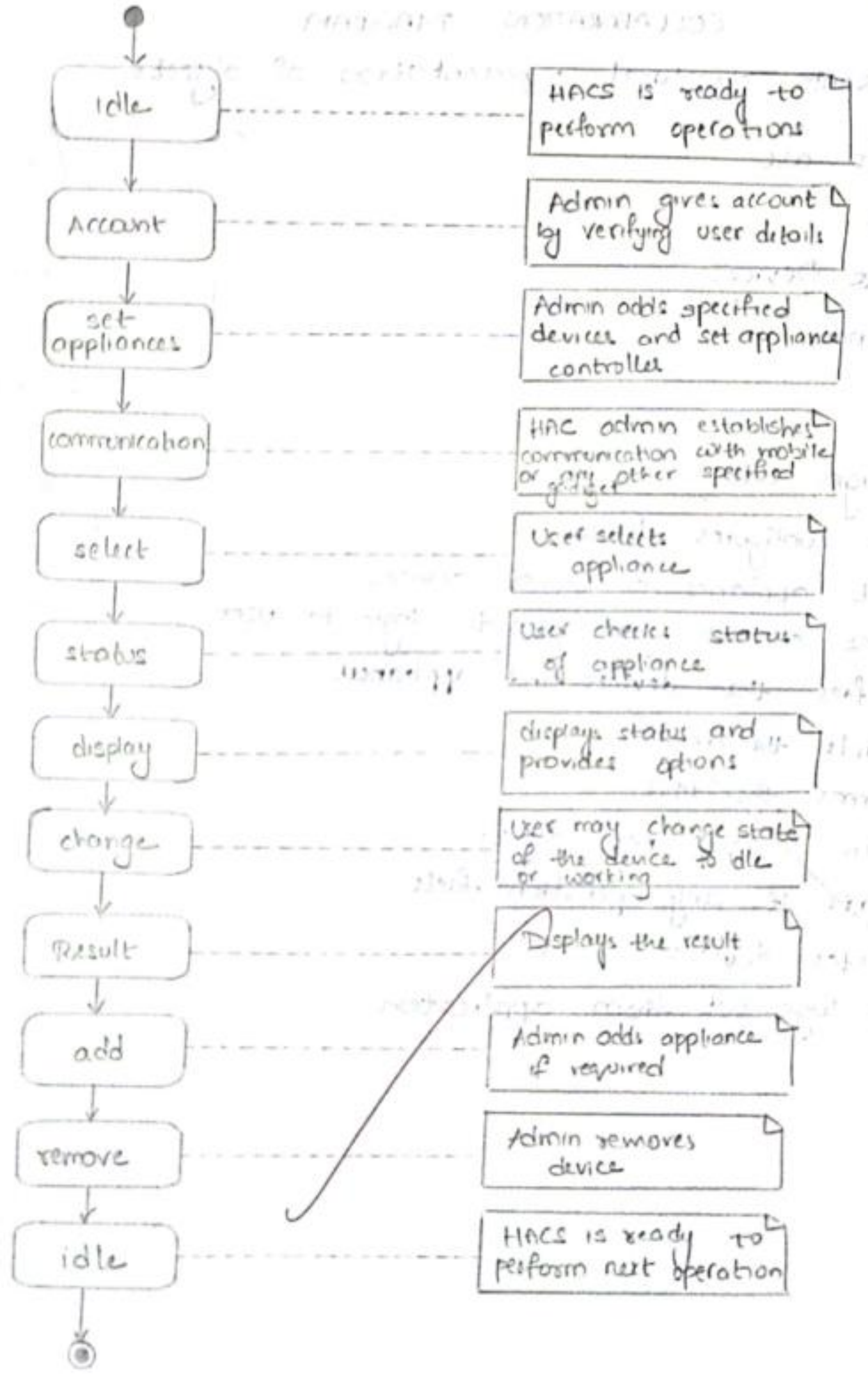
The Objects are

- Admin
- Remote device
- Appliance
- User

The Messages are

1. Admin configures device
2. Connects appliance to remote device
3. Creates account and provides login to user
4. Specifies the devices and appliances
5. Controls the device
6. Performs the task
7. Admin connects new devices
8. Report if any operation fails
9. Deletes devices
10. User logs out from application

Statechart Diagram


idle
HACS is ready to perform operations
Account
Admin gives account by verifying user details
set appliances
Admin adds specified devices and set appliance controller
communication
HAC admin establishes communication with mobile or any other specified gadget
select
User selects appliance
status
User checks status of appliance
display
displays status and provides options
change
User may change state of the device to idle or working
Result
Displays the result
add
Admin adds appliance if required
remove
Admin removes device
idle
HACS is ready to perform next operation

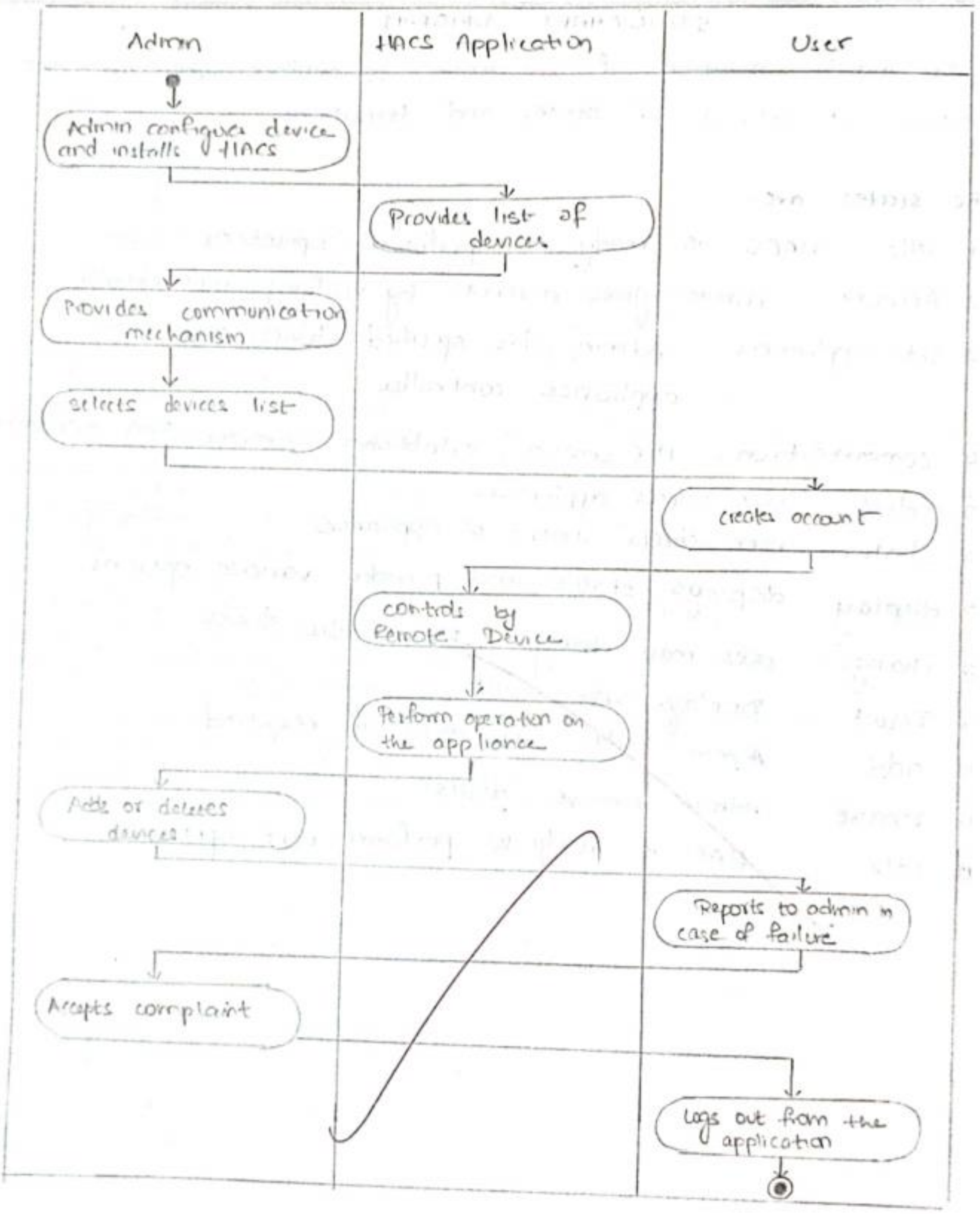
STATECHART DIAGRAM

- It depicts movement of one state to another when an event occurs. It consists of states and transitions.

The States are

1. idle – HACS is ready to perform operations
2. account – Admin gives account by verifying user details
3. set appliances – Admin adds specified devices and sets appliance controller
4. communication – HACS admin establishes communication mechanism
5. select – User selects appliance
6. status – User checks status of appliance
7. display – Displays status and provides various options
8. change – User may change state of the device
9. result – Displays the result
10. add – Admin adds appliances if required
11. remove – Admin removes device
12. idle – HACS is ready to perform next operation

Activity Diagram


Admin
HACS Application
User
Admin configures device and installs HACS
Provides list of devices
Provides communication mechanism
selects devices list
creates account
controls by Remote Device
Perform operation on the appliance
Adds or deletes devices
Reports to admin in case of failure
Accepts complaint
Logs out from the application

ACTIVITY DIAGRAM

- It depicts flow from one activity to other.

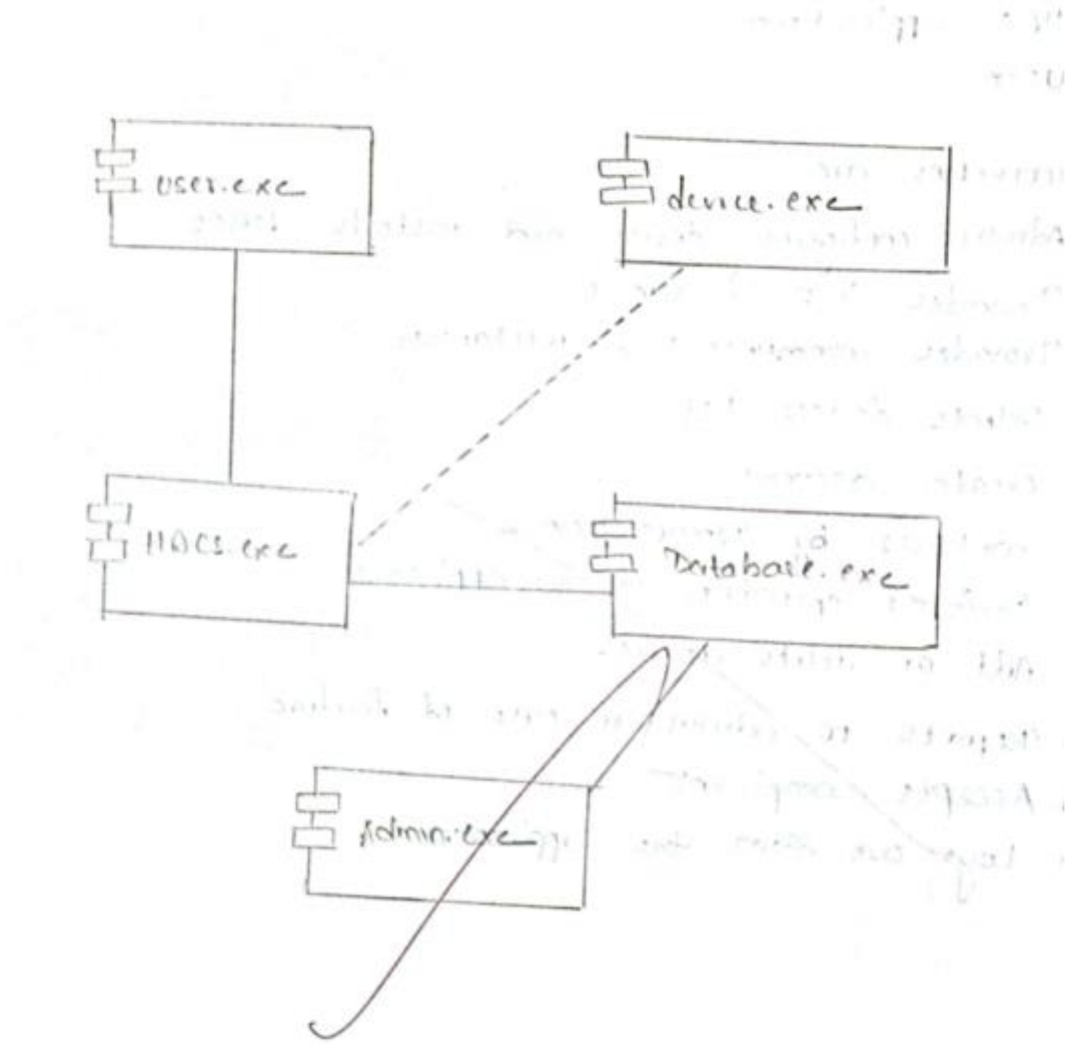
The Swimlanes are

- Admin
- HACS application
- User

The Activities are

1. Admin configures device and installs HACS
2. Provides list of devices
3. Provides communication mechanism
4. Selects devices list
5. Creates account
6. Controls by Remote device
7. Performs operation on the appliance
8. Add or delete devices
9. Reports to admin in case of failure
10. Accepts complaint
11. Logs out from the application

Component Diagram



COMPONENT DIAGRAM

- It gives implementation details of the system.

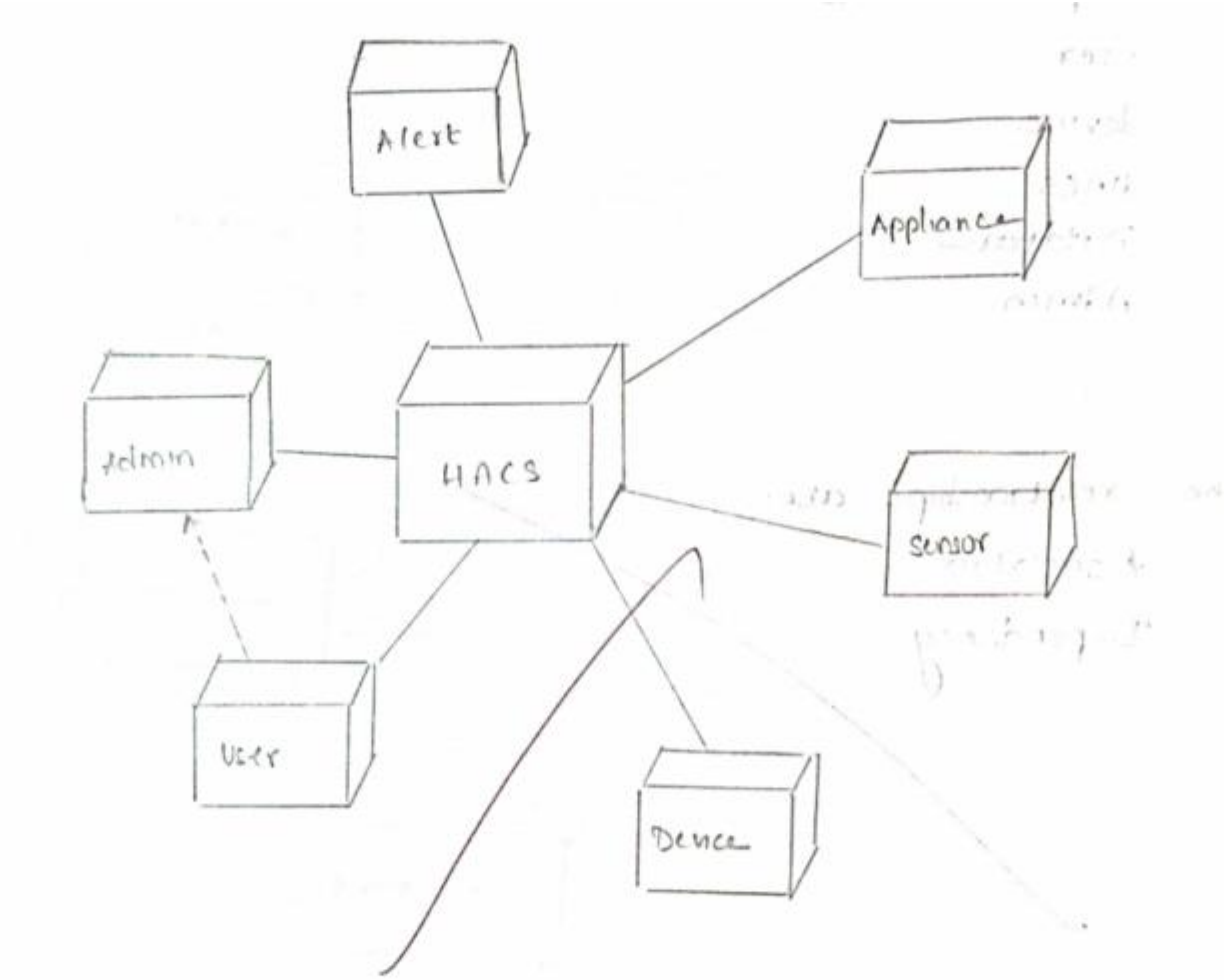
The Components are

- User
- Device
- HACS
- Database
- Admin

The Relationships are

- Association
- Dependency

Deployment Diagram



DEPLOYMENT DIAGRAM

- It consists of nodes to deploy the application.

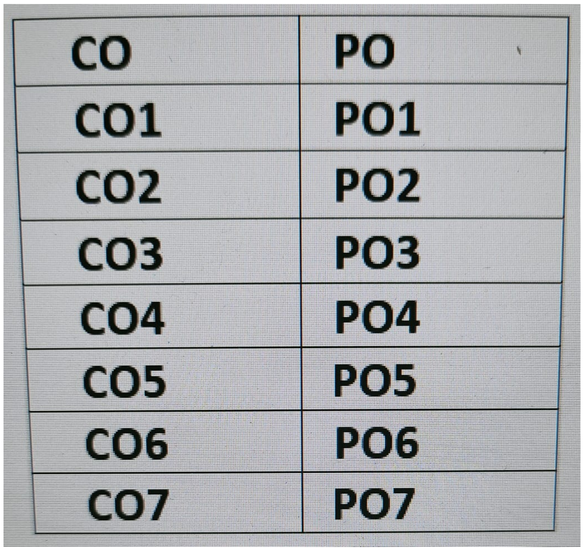
The Nodes are

- HACS
- Alert
- Appliance
- Admin
- Sensor
- User
- Device

The Relationships are

- Dependency
- Association

Result : Thus all the views of Home Alliance control System has been modelled and it has been mapped to following cos and pos.



CO	PO
CO1	PO1
CO2	PO2
CO3	PO3
CO4	PO4
CO5	PO5
CO6	PO6
CO7	PO7